NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

DISSERTATION

LAYERED-BASED CODING, SMOOTHING, AND SCHEDULING OF LOW-BIT-RATE VIDEO FOR TELECONFERENCING OVER TACTICAL ATM NETWORKS

by

Robert E. Parker, Jr.

September 1999

Dissertation Supervisor: **Murali Tummala**

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Ph.D. Dissertation

4. TITLE AND SUBTITLE
LAYER-BASED CODING, SMOOTHING, AND SCHEDULING OF LOW-BIT-RATE VIDEO FOR TELECONFERENCING OVER TACTICAL ATM NETWORKS

5. FUNDING NUMBERS

6. AUTHOR(S)
Parker, Jr., Robert E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This work investigates issues related to distribution of low-bit-rate video within the context of a teleconferencing application deployed over a tactical ATM network. The main objective is to develop mechanisms that support transmission of low bit rate video streams as a series of scalable layers that progressively improve quality. The hierarchical nature of the layered video stream is actively exploited along the transmission path from the sender to the recipients to facilitate transmission.

A new layered coder design tailored to video teleconferencing in the tactical environment is proposed. Macroblocks selected due to scene motion are layered via subband decomposition using the fast Haar transform. A generalized layering scheme groups the subbands to form an arbitrary number of layers. As a layering scheme suitable for low-motion video is unsuitable for static slides, the coder adapts the layering scheme to the video content. A suboptimal rate control mechanism that reduces the k-dimensional rate-distortion problem resulting from the use of multiple quantizers tailored to each layer to a 1-dimensional problem by creating a single rate-distortion curve for the coder in terms of a suboptimal set of k-dimensional quantizer vectors is investigated. Rate control is thus simplified into a table lookup of a codebook containing the suboptimal quantizer vectors. The rate controller is ideal for real-time video and limits fluctuations in the bit-stream with no corresponding visible fluctuations in perceptual quality.

A traffic smoother prior to network entry is developed to increase queuing and scheduler efficiency. Three levels of smoothing are studied: frame, layer, and cell interarrival. Frame level smoothing occurs via rate control at the application. Interleaving and cell

14. SUBJECT TERMS
Low-bit-rate Video Coding, Rate Control, Scheduling, Traffic Smoothing

15. NUMBER OF PAGES
255

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18

interarrival smoothing are accomplished using a leaky bucket mechanism inserted prior to the adaptation layer or within the adaptation layer. Simulations indicate that smoothing lowers bandwidth requirements for a given quality of service and that interleaving cells from different layers enhances the effectiveness of priority-based scheduling schemes.

A new cell-scheduling scheme is proposed that exploits the layered video hierarchy to allow more graceful degradation in visual quality during periods of cell loss. Quality of service at the connection level is maintained using an optimal scheduling algorithm that accounts for the cell loss rate and cell transfer delay requirements for each connection. Within the connection, a prioritization scheme denies service to cells from lower priority layers during periods of congestion and cells deemed non-viable due to group of blocks (GOB) corruption to increase the probability that cells from higher priority layers are transmitted. Simulations indicate that protecting higher priority layers requires accepting a corresponding decrease in throughput. Depending on the prioritization scheme used, cell loss rates for the base video layer can either be maintained at the desired rate or improved by an order of magnitude relative to no prioritization. Cell discarding allows the scheduler to recover bandwidth from non-viable cells although the impact within the connection depends on the service discipline. As the GOB size increases, cell discarding is improved if cells from different layers are interleaved to reflect spatial dependency between the base layer and the enhancement layers.

Approved for public release; distribution is unlimited.

LAYER-BASED CODING, SMOOTHING, AND SCHEDULING OF LOW-BIT-RATE VIDEO FOR TELECONFERENCING OVER TACTICAL ATM NETWORKS

Robert E. Parker, Jr.
Lieutenant Commander, United States Navy
B.S., North Carolina State University, 1985
M.S., Naval Postgraduate School, 1992

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

September 1999

Author: _____
Robert E. Parker, Jr.

Approved by: _____
Roberto Cristi
Associate Professor of Electrical
and Computer Engineering

_____
David C. Jenn
Associate Professor of Electrical
and Computer Engineering

_____
John C. McEachen II
Assistant Professor of Electrical
and Computer Engineering

_____
C. Thomas Wu
Associate Professor of
Computer Science

_____
Murali Tummala
Professor of Electrical and Computer Engineering
Dissertation Advisor and Committee Chair

Approved by: _____
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

Approved by: _____
Anthony Ciavarelli, Associate Provost for Instruction

ABSTRACT

This work investigates issues related to distribution of low-bit-rate video within the context of a teleconferencing application deployed over a tactical ATM network. The main objective is to develop mechanisms that support transmission of low bit rate video streams as a series of scalable layers that progressively improve quality. The hierarchical nature of the layered video stream is actively exploited along the transmission path from the sender to the recipients to facilitate transmission.

A new layered coder design tailored to video teleconferencing in the tactical environment is proposed. Macroblocks selected due to scene motion are layered via subband decomposition using the fast Haar transform. A generalized layering scheme groups the subbands to form an arbitrary number of layers. As a layering scheme suitable for low-motion video is unsuitable for static slides, the coder adapts the layering scheme to the video content. A suboptimal rate control mechanism that reduces the k-dimensional rate-distortion problem resulting from the use of multiple quantizers tailored to each layer to a 1-dimensional problem by creating a single rate-distortion curve for the coder in terms of a suboptimal set of k-dimensional quantizer vectors is investigated. Rate control is thus simplified into a table lookup of a codebook containing the suboptimal quantizer vectors. The rate controller is ideal for real-time video and limits fluctuations in the bit-stream with no corresponding visible fluctuations in perceptual quality.

A traffic smoother prior to network entry is developed to increase queuing and scheduler efficiency. Three levels of smoothing are studied: frame, layer, and cell interarrival. Frame level smoothing occurs via rate control at the application. Interleaving and cell interarrival smoothing are accomplished using a leaky bucket mechanism inserted prior to the adaptation layer or within the adaptation layer. Simulations indicate that smoothing lowers bandwidth requirements for a given quality of service and that interleaving cells from different layers enhances the effectiveness of priority-based scheduling schemes.

A new cell-scheduling scheme is proposed that exploits the layered video hierarchy to allow more graceful degradation in visual quality during periods of cell loss. Quality of service at the connection level is maintained using an optimal scheduling algorithm that accounts for the cell loss rate and cell transfer delay requirements for each connection. Within the connection, a prioritization scheme denies service to cells from lower priority layers during periods of congestion and cells deemed non-viable due to group of blocks (GOB) corruption to increase the probability that cells from higher priority layers are transmitted. Simulations indicate that protecting higher priority layers requires accepting a corresponding decrease in throughput. Depending on the prioritization scheme used, cell loss rates for the base video layer can either be maintained at the desired rate or improved by an order of magnitude relative to no prioritization. Cell discarding allows the scheduler to recover bandwidth from non-viable cells although the impact within the connection depends on the service discipline. As the GOB size increases, cell discarding is improved if cells from different layers are interleaved to reflect spatial dependency between the base layer and the enhancement layers.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

1. IT-21 2
2. Video Teleconferencing 2
3. Multimedia Applications and QoS 3

B. PROBLEM SCENARIO 4

1. Target Scenario 4
2. Video Compression and Robustness 6

C. DISSERTATION OBJECTIVES 8

D. DISSERTATION ORGANIZATION 13

II. NETWORK ARCHITECTURES FOR MULTIMEDIA TRAFFIC 15

A. LEGACY IP-BASED NETWORK 16

1. IP and Multicast IP 17
2. Transport Layer Protocols 18
3. Real-time Transport Protocol 18
4. Suitability of RTP/IP for VTC 20

B. ATM NETWORKS 20

1. ATM Protocol Model 21
2. Logical Connections 23
3. ATM Cell Format 24
4. ATM Service Classes 25
5. ATM Adaptation Layer (AAL) 28
6. ATM Multicast 31

C. WIRELESS NETWORKS 33

1. Logical Link Control 34
2. Medium Access Control 35
3. Physical Layer 35

D. LAYERED VTC OVER ATM 36

1. ITU-T Multimedia Standards 36
2. Layered Video Considerations 37
3. Multiple VCC Case 40
4. Single VCC Case 43

III. VIDEO CODING TECHNIQUES 49

A. VIDEO COMPRESSION OVERVIEW 49

B. VIDEO CODING HIERARCHY 51

C. INTRAFRAME CODING 52

1. Transform Coding 53
2. Scalar Quantization 56
3. Entropy Encoding 58
4. Quality of Reproduced Video 59

D. INTERFRAME CODING 59

1. Block Updating 60
2. Differential Pulse Code Modulation 61
3. Forward Motion-Compensated Prediction 61
4. Bi-directional Motion Compensation 62
5. Distance Metrics 63
6. Hybrid Video Coding 64

E. ERROR ROBUSTNESS 65

1. Motion JPEG 66
2. MPEG 66
3. H.263 67
4. Error Propagation 67

F. SUBBAND AND WAVELET CODING 68

IV. LOW-COMPLEXITY LAYERED VIDEO CODING 73

A. BACKGROUND 74

B. LAYERED VIDEO CODING 78

1. Progressive JPEG Encoding 79
2. Pyramid Coding 81
3. Wavelet and Subband Coding 83

C. A LOW-COMPLEXITY ADAPTIVE LAYERED CODER DESIGN 86

1. Block Selection for Motion Compensation 87
2. Aging Algorithm 91
3. Layering Strategy 94
4. Quantization and Lossless Coding 107

D. RESULTS 109

E. SIMPLE LAYERED-VIDEO RATE CONTROL 113

1. A Rate-Distortion Approach 115
2. Approximation of the 3-D Rate-Distortion Curve 118

V. TRAFFIC SMOOTHING 129

A. INTRODUCTION 129

B. VIDEO TRAFFIC MODELING 134

1. Autoregressive Models 135
2. Histogram-based Traffic Modeling 136
3. Determining Model Parameters 138
4. Queuing Analysis 140
5. Application 143

C. SMOOTHING LAYERED VIDEO TRAFFIC 143

1. Cell Level Traffic Smoothing 144
2. Predictive Smoothing 146
3. Interleaving/Transmission Order 148
4. Smoothing: Single VCI Case 152
5. Smoothing: Multiple VCI Case 154
6. Smoothing Results 157

VI. SCHEDULING LAYERED VIDEO TRAFFIC 161

A. SCHEDULING CRITERIA 162

B. QOS SCHEDULING ALGORITHMS 164

1. Reservation Schemes 165
2. STE and BCLPR 166
3. STEBR 168

C. LAYERED SCHEDULING 169

1. QoS Filtering and the STEBR Algorithm 169
2. GOB Dropping 174

D. RESULTS 179

VII. CONCLUSIONS 191

A. SUMMARY OF WORK 191

B. SUGGESTIONS FOR FUTURE RESEARCH 193

APPENDIX A. OPNET MODEL CODE 197

A. LAYERED VIDEO SCHEDULER 197

1. Header Block 198
2. State Variable Block 198
3. Temporary Variable Block 199
4. Function Block 200
5. Init State 202
6. Arrival State 203
7. SVC_Start State 205
8. SVC_Complete State 209

B. LAYERED VIDEO SOURCE 209

1. Header Block 210
2. State Variable Block 211
3. Temporary Variable Block 211
4. Function Block 211
5. Init State 212
6. Transition State 213
7. Send_cell State 214

APPENDIX B. MMRP MODEL PARAMETERS 217

LIST OF REFERENCES 221

INITIAL DISTRIBUTION LIST 231

LIST OF FIGURES

Figure I.1: Simple VTC Multicast with Two Active and Two Passive Nodes. 3
Figure I.2: Hybrid ATM Wireline/Wireless Network. 6
Figure I.3: Functional Diagram for a Multicast VTC Application 10
Figure II.1: IP-Based Network Protocol Stack for Real-time Traffic. 17
Figure II.2: ATM Protocol Architecture [23] 22
Figure II.3: ATM Network Configuration [27] 23
Figure II.4: ATM Cell Format at the UNI [23]. 24
Figure II.5: Bandwidth Allocation for ATM Service Categories [18]. 27
Figure II.6: Segmentation at the AAL [18]. 29
Figure II.7: ATM Point-to-multipoint Multicast. 32
Figure II.8: DLC for a Packet-Based Radio Network. 34
Figure II.9: ITU-T H.310 B-ISDN Terminal. 37
Figure II.10: Transmitting Layered Video Using AAL5 and Multiple VCCs. 41
Figure II.11: Use of the SDU Bit to Locate Application PDU Boundaries with AAL5. .. 42
Figure II.12: Transmitting Layered Video Using AAL2 and a Single VCC. 44
Figure II.13: Identifying Application PDUs in a Multiplexed Cell Flow. 47
Figure III.1: Organizational Hierarchy for Compressed Video. 52
Figure III.2: Overview of the Steps Comprising Intraframe Coding. 53
Figure III.3: Frequency Interpretation of DCT Coefficients. 55
Figure III.4: Structural Decomposition of Image Elements [6]. 55
Figure III.5: Hybrid Video Coder with Motion Compensation and DCT-based Compression. 64
Figure III.6: Typical GOP, $N = 9$, $M = 3$. 67
Figure III.7: DWT-based Image Compression. 70
Figure III.8: Octave-band Decomposition. 71
Figure IV.1: Video Transmission over a Heterogeneous Network from [45]. 74
Figure IV.2: Video Transmission Using RLM. 77
Figure IV.3: Overview of Layered Video Coding/Decoding. 78

Figure IV.4: Basic Layered Video Coder Using Wavelets. 84
Figure IV.5: Functional Block Diagram of the Hybrid FHT/DCT Layered Coder. 87
Figure IV.6: Block Search Order: a) Clockwise Search and b) Cross-pattern Search. 88
Figure IV.7: Comparison of ASD and SAD for Block Selection. 90
Figure IV.8: Pseudocode for Aging Algorithm. 93
Figure IV.9: Splitting a Macroblock into Uniform Subbands. 98
Figure IV.10: Partitions Resulting from Merge Algorithm. 100
Figure IV.11: Final Layering Scheme for Motion Video Sequences. 103
Figure IV.12: Partitions Resulting from Merge Algorithm (Slide Sequence). 105
Figure IV.13: Final Layering Scheme for Static Slide Sequences. 105
Figure IV.14: Partitions Remaining After Merging Similar Non-Contiguous Partitions. 106
Figure IV.15: Partitions for the Purpose of Quantization. 106
Figure IV.16: Quantization and Coding for Motion Video Macroblocks. 107
Figure IV.17: Quantization and Coding for Static Macroblocks. 109
Figure IV.18: Original and Reconstructed Frames From a Motion Video Sequence. 110
Figure IV.19: Original and Reconstructed Frames From a Static Video Sequence. 111
Figure IV.20: Bitrate per Frame for the Layered Video Sequence. 112
Figure IV.21: Reconstructed pSNR for the Layered Video Sequence. 112
Figure IV.22: Rate-Distortion Curve and a Possible Optimal Solution. 114
Figure IV.23: Operational Rate-Distortion Curve for Motion Video. 119
Figure IV.24: Quantizer Table Triplet Values for Motion Video. 119
Figure IV.25: Operational Rate Control Curve. 120
Figure IV.26: Bit Rate Traces for a) Controlled and b) Uncontrolled Video Sequences 122
Figure IV.27: pSNR Variation for a) Controlled and b) Uncontrolled Video Sequences 124
Figure IV.28: Operational Rate-Distortion Curve for Static Sequences. 127
Figure V.1: Leaky Bucket Access Mechanism. 133

Figure V.2: Video Trace for a Low Activity Sequence. 135
Figure V.3: Minisource Video Model. 136
Figure V.4: Markov-modulated Poisson Process (MMPP). 138
Figure V.5: Cell and Burst Regions for Cell Loss. 142
Figure V.6: Cell Level Traffic Smoothing. 145
Figure V.7: Simulated and Predicted VBR VTC Sequences Using RLS. 148
Figure V.8: Prediction Error for 3-Tap Filter Using RLS. 148
Figure V.9: Equivalent Cell Flows for Prioritized Traffic. 150
Figure V.10: Several Possible Interleaving Schemes Given a 4:2:2 Ratio Among Layers. 150
Figure V.11: Cell Smoothing and Cell Interleaving Over One Frame 151
Figure V.12: Smoothing at the Application with a Single VCI. 153
Figure V.13: Smoothing Within the AAL with a Single VCI. 154
Figure V.14: Smoothing at the Application with Multiple VCIs. 156
Figure V.15: Smoothing Within the AAL with Multiple VCIs. 157
Figure V.16: Estimated CLR for Rate-controlled and Non-rate-controlled VTC Traffic. 158
Figure V.17: Estimated CLR for Rate-controlled and Non-rate-controlled VTC Traffic (3 sources). 158
Figure V.18: Deterministic and Poisson Intraframe Smoothing. 159
Figure V.19: Effect of Cell Concatenation of Scheduler Performance. 160
Figure VI.1: A Switch Controlling Access to a Network 161
Figure VI.2: Admission Regions for FCFS and an Optimal Scheduling Policy. 163
Figure VI.3: Scheduling Different Priority or QoS Classes. 165
Figure VI.4: Cost Filtering per Layer for a) Bandwidth Sharing and b) Priority Sharing. 172
Figure VI.5: Modified STEBR Algorithm. 173
Figure VI.6: The Effect of Cell Discard on a GOB 175
Figure VI.7: Competition Between Usable and Unusable Cells. 175

Figure VI.8: Discard Policy Following a Cell Loss from: a) Base Layer GOB or b) Enhancement Layer GOB 177
Figure VI.9: Interleaving Layers Cells to Minimize Information Loss. 178
Figure VI.10: Partial GOB Dropping Algorithm. 179
Figure VI.11: Network Scenario. 180
Figure VI.12: CLR for Bandwidth Sharing and Service Deferrals. 181
Figure VI.13: CLR for Bandwidth Sharing and No Service Deferrals. 181
Figure VI.14: CLR for Priority Sharing and No Service Deferrals 182
Figure VI.15: Throughput under Each Scheduling Scheme 183
Figure VI.16: CLR for Bandwidth Sharing, Service Deferrals, and GOB Dropping. 183
Figure VI.17: CLR for Bandwidth Sharing, No Service Deferrals, and GOB Dropping. 184
Figure VI.18: Throughput Variation with Partial GOB Dropping. 185
Figure VI.19: Cell Arrangements Considered for a 4:2:2 Bit Allocation: a) Concatenated or b) Interleaved 186
Figure VI.20: Relative Affect of Interleaving and Concatenating on CLR with Bandwidth Sharing and Service Deferrals. 187
Figure VI.21: Relative Affect of Interleaving and Concatenating on CLR with Bandwidth Sharing and Without Service Deferrals. 188
Figure A.1: Finite State Machine for Scheduler Process Model. 198
Figure A.2: Finite State Machine for a Layered Video Traffic Model. 210
Figure B.1: Rate-controlled VBR Video Sequence 217
Figure B.2: Predicted and Actual Histograms 218
Figure B.3: Actual and Predicted Autocorrelation Functions. 219

LIST OF TABLES

Table I.1: Tactical VTC Multimedia Requirements.....6
Table II.1: ATM Service Classes [28]. 26
Table II.2: AAL Protocol Mapping to Service Classes [18]. 30
Table II.3: ATM Cell-Tagging Scheme for Layered Video.....46
Table III.1: Error Propagation in Popular Video Codecs. 68
Table IV.1: Resolutions Supported in Gharavi and Partovi's Layered Coder. 83
Table IV.2: Significance and Determination of Wavelet Subbands. 95
Table IV.3: Preliminary Layer Assignments.....96
Table IV.4: Subband Variances after a First Order Decomposition (Motion Video). 99
Table IV.5: Subband Variances after a Second Order Decomposition (Motion Video)...99
Table IV.6: Subband Variances after a First Level Decomposition (Slide Sequence). ..104
Table IV.7: Subband Variances after a Second Level Decomposition
(Slide Sequence). 104
Table IV.8: Scan Order for Encoding Quantized Coefficients. 108
Table IV.9: Rate Controlled and Uncontrolled Sequence Statistics. 124
Table B.1: State Probabilities and Arrival Rates for Quantized Video Source.....217
Table B.2: Infinitesimal Generating Function for Quantized Video Source.....218

SYMBOLS AND NOTATIONS

$A[i]$	arrivals from connection i
β	bit rate control parameter
$\overline{B}$	average bit allocation per frame
B_i	bit allocation for frame I
$B_{i,j}$	bit allocation for frame i, macroblock j
C_L	line capacity
ΔB_{inter}	bit allocation error (or deviation) in successive frames
ΔB_{intra}	bit allocation error (or deviation) within a frame
Δ_i	cost increment for connection i
ΔQ_i	change in quantizer table entry for frame i
$\Delta Q_{i,j}$	change in quantizer table entry for frame i, macroblock j
$\Delta\sigma^2$	variance distance metric
D	distortion
D_i	distortion in layer i
$D(R)$	distortion as a function of bit rate
$DS[i]$	cells denied service from connection I
e_{ASD}	ASD distance metric
e_{MSE}	MSE distance metric
e_{SAD}	SAD distance metric
f	frame rate
$f(i,j)$	pixel intensity
$F(u,v)$, F_{uv}	DCT coefficient
F_{quv}	quantized DCT coefficient
HH	subband via first order FHT analysis; retains diagonal detail of image
HL	subband via first order FHT analysis; retains vertical edge detail of image
J	Lagrangian cost function
k_i	subband i
K	maximum peak-to-peak pixel intensity
L	layer
λ	Lagrange multiplier
λ_I	arrival rate for state i
LH	detail matrix via first order FHT analysis; retains horizontal edge detail
LL	average matrix via first order FHT analysis; retains lowpass content
μ	service rate
M	infinitesimal generating function
n_{MB}	current macroblock
N_b	total number of subbands created by recursive application of FHT
N_{MB_i}	total number of macroblocks in frame i
$\overline{N}_{MB}$	average number of macroblocks chosen per frame in test video sequences

π	steady-state distribution vector
π_i	steady-state probability for state i
P	partition comprised of one or more subbands
P	state transition matrix
P_L	system loss probability
P_{L_i}	system loss probability for state i
q_1	first quantizer parameter
q_2	second quantizer parameter
q_3	third quantizer parameter
Q	quantizer
Q_i	quantizer step size for frame i
Q_{mb}	quantizer step size for the current macroblock
Q_{uv}	quantizer matrix step size entry
ρ	network load
r	token generation rate
R	bit rate
R_c	bit rate constraint
R_{target}	target bit rate
σ^2	variance
$\sigma_{k_i}^2$	variance for subband k_i
$\sigma_{L_i}^2$	variance for layer L_i
σ^2_{max}	maximum variance of all subbands
σ^2_{min}	minimum variance of all subbands
$\sigma_{P_k}^2$	variance for partition P_k
T	state sojourn time
T_k	service slot k
W_j	weight for bit allocation assigned to macroblock j
x	original image
$\hat{x}$	reconstructed image
x_0^a	original data vector prior to FHT analysis
x_1^a	first order FHT average analysis vector
x_1^d	first order FHT detail analysis vector
$x_{n,m}$	pixel value in current block
$x^R_{n,m}$	pixel value in reference block

ACRONYMS AND ABBREVIATIONS

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ASD	Absolute Sum of Differences
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
BCLPR	Balanced-CLP-Ratio Scheduling Algorithm
BER	Bit Error Rate
bpf	Bits per Frame
bpp	Bits per Pixel
CBR	Constant Bit Rate
CELP	Code Excited Linear Prediction
CID	Channel Identifier
CIF	Common Intermediate Format
CLP	Cell Loss Priority
CLP	Cell Loss Probability
CLR	Cell Loss Rate
COTS	Commercial Off-The-Shelf Technology
CPCS	Common Part Convergence Sublayer
CS	Convergence Sublayer
D/D/1/*K*	Deterministic Arrivals and Service, Single Server, Finite Queue
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DWT	Discrete Wavelet Transform
EOB	End-Of-Block
FCFS	First-come, First-serve
FHT	Fast Haar Transform
fps	Frames per Second
GOB	Group of Macroblocks or Blocks
GOP	Group of Pictures
HVS	Human Visual System
ICLP	Instantaneous Cell Loss Probability
IDCT	Inverse Discrete Cosine Transform
IP	Internet Protocol
IT-21	Information Technology for the 21st Century
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardization Sector
JPEG	Joint Photographic Experts Group
kbps	Kilobits per Second
KLT	Karhunen-Loeve Transform
LAN	Local Area Network
maxCTD	Maximum Cell Transfer Delay
MB	Macroblock

MBone	Multicast Backbone
M-JPEG	Motion-JPEG
MMDS	Multichannel Multipoint Distribution Service
MMPP	Markov-modulated Poisson Process
MMRP	Markov-modulated Rate Process
MPEG	Moving Pictures Experts Group
MSE	Mean Squared Error
NNI	Network-network Interface
PCM	Pulse Code Modulation
PCR	Peak Cell Rate
PDU	Protocol Data Unit
pSNR	peak Signal-to-Noise Ratio
PSTN	Public Switched Telephone Network
PT	Payload Type
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
RLE	Run-Length Encoding
RLM	Receiver-Based Layered Multicast
RSVP	Resource ReSerVation Protocol
RTCP	Real-time Control Protocol
RTP	Real-time Transport Protocol
SAD	Sum of Absolute Differences
SAP	Service Access Point
SAR	Segmentation and Reassembly Sublayer
SCR	Sustainable Cell Rate
SDU	Service Data Unit
SNR	Signal-to-Noise Ratio
SSCS	Service-specific Convergence Sublayer
STE	Shortest Time to Expiration Scheduling Algorithm
STEBR	STE with BCLPR Scheduling Algorithm
TCP	Transmission Control Protocol
ToE	Time to Expiration
UBR	Unassigned Bit Rate
UD	Uniquely Decipherable
UDP	User Datagram Protocol
UNI	User-network Interface
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VLC	Variable-Length Coding
VOD	Video-On-Demand
VPC	Virtual Path Connection
VPI	Virtual Path Identifier
VTC	Video Teleconferencing

ACKNOWLEDGEMENT

I dedicate this work to my family, my wife Cheryl and my children, Alex and Sam. Without Cheryl's omnipresent support and understanding, completing this work would have been far more difficult and less gratifying. Words cannot express the gratitude I have for her patience, support, and love. Alex and Sam have continually kept my life interesting; my life is that much richer for their presence.

My eternal thanks to my adviser, Prof. Murali Tummala, for guiding me through the painful process of discovering and exploring my research topic. Three years is a long period of time and there are numerous hurdles to completing Ph.D. studies. His continual interest, support, and advice were of immense value in helping me past each hurdle, from qualification to his careful and exacting editing of my dissertation. I deeply value his friendship, the interest he has taken in my advancement, and the opportunities he had afforded in pursuing my educational interests.

Finally, I have to give many thanks to Steve Skretkowicz. His programming efforts were crucial to facilitating my layered coding research and allowing pursuit of ideas I would have otherwise put aside. He dedicated probably too many hours to bring my ideas to life, especially my bad ideas, but his enthusiasm was always infectious.

I. INTRODUCTION

Multimedia applications support the processing, transmission, and control of streams of related audio-visual signals including text, images, audio, and video data [1]. Common examples include streaming applications, such as video-on-demand (VOD), and interactive applications, such as video and audio teleconferencing. Multimedia applications offer difficult challenges for network design due to the need to bound data loss in transmission, the need to limit transmission delays, and the need for synchronizing the related streams comprising a multimedia session. In particular, video teleconferencing (VTC) demonstrates the great potential of multimedia applications to deliver information but, at the same time, poses difficult distribution problems for the hosting network.

VTC plays an important role in the U.S. Navy's Information Technology for the 21st Century initiative (IT-21). IT-21 seeks to transform the current platform centric approach to warfighting to a network centric approach that leverages information superiority with current and planned smart weapons [2]. At the battlegroup level, deploying VTC over a tactical network that links individual units via a wireless link offers several benefits including collaborative planning, remote maintenance, distance learning, and telemedicine. However, a tactical network thus envisioned present constraints not typically present in traditional wireline networks. The tactical network may be viewed as an internetwork of shipboard wireline local area networks (LANs) interconnected by a wireless channel. The wireless channel serves as a bottleneck within the tactical network and constrains both the available bit rate and transmission quality. Each of these constraints impacts the perceived quality of any deployed VTC application.

A. BACKGROUND

This section provides additional information on the IT-21 initiative and VTC to provide a context for the problem scenario in the next section. Additionally, the type of service required to support VTC is briefly considered.

1. IT-21

A brief examination of the IT-21 initiative is valuable for determining the baseline network architecture to host a tactical VTC application. The goal of IT-21 is to link all U.S. Forces together in a network that enables the transmission of voice, video and data from individual workstations seamlessly to both local and remote users [2][4]. The anticipated network is heterogeneous and allows connectivity among wireline LANs using both wireless and satellite communication links. All networks and interfaces are to use commercial off-the-shelf (COTS) technology built to current industry standards.

Focusing on the battlegroup level, shipboard LANs are to have ATM backbones. Individual workstation connectivity is provided initially via 100 Mbps Fast Ethernet with a future transition to direct ATM connections. Connectivity among units of the battlegroup is provided by EHF links with a minimum data rate of 128 kbps to support messaging and maintain a common tactical picture. However, to support multimedia applications, such as VTC or collaborative planning with high resolution, early projections indicate that a minimum data rate of 1280 kbps is required.

2. Video Teleconferencing

Teleconferencing systems can be broken into three categories: audio-only, audio and graphics, and video. VTC is an interactive application requiring low network latency, bounded delay jitter, and low cell loss to both preserve audiovisual quality and maintain the sense of interactivity. In addition, careful synchronization is required between the audio and video streams. While communication may be unicast as in peer-to-peer applications, the more challenging problem of multicast communication is considered here. As such, each sender is assumed to transmit to multiple receivers in the multicast group. In turn, the multicast group consists of some combination of active participants that receive and transmit and passive participants that receive only. This situation is illustrated in Figure I.1.

Since video, and audio to a lesser degree, is bandwidth intensive, signals are compressed prior to transmission and trade some reduction in quality for a reduction in bandwidth. Multimedia communications, therefore, require dedicated terminals, which

capture and prepare signals for transmission over the network and reconstruct received streams by decompressing and resynchronizing different streams as required. Commercial VTC applications have been facilitated by the emergence of ITU standards for multimedia terminals [3]. Each standard targets a bandwidth range (and thus quality), a particular networking standard, and incorporates a family of associated standards to support the required audio and video compression, control signals, and network interface.

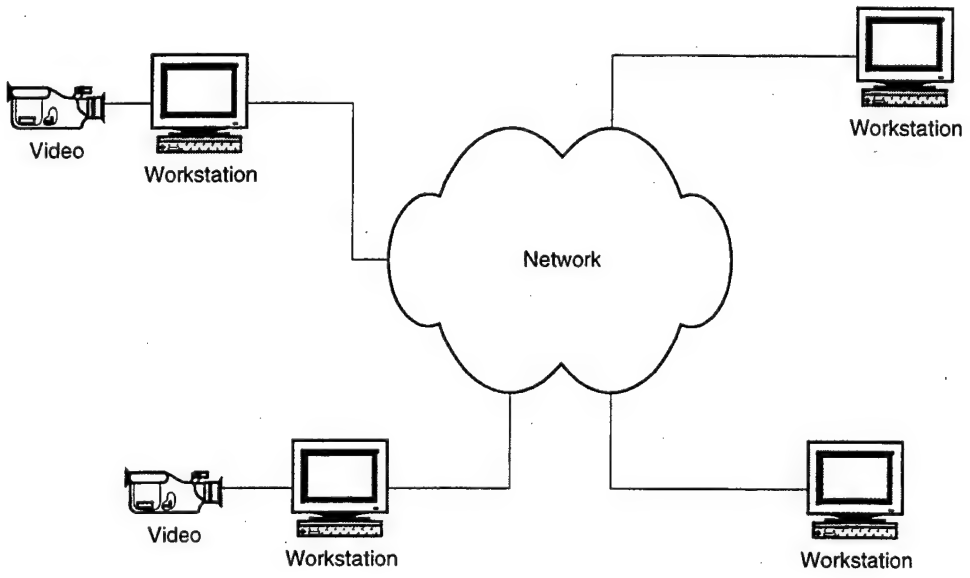

Figure I.1: Simple VTC Multicast with Two Active and Two Passive Nodes.

3. Multimedia Applications and QoS

Quality of service (QoS) denotes a set of one or more parameters describing the level of service granted to an application by a network or required from the network by the application for acceptable performance. Many possible QoS parameters exist, but the typical parameters employed are maximum allowable delay, delay variation or jitter, and cell loss rates. The QoS requirements for a particular multimedia application depend on the types of information transmitted and the manner in which the information is compressed or packaged for transmission. More generally, multimedia applications are characterized by the manner in which information is distributed, the degree of interactivity, and the type of information transported [1].

Multimedia communications are either unicast or multicast. Unicast represents peer-to-peer communication while multicast represents *m*-to-*n* communication, where *m* ranges from 1 to *n*. Unicast examples include client-server applications, such as VOD. Multicast examples include distance-learning and tele-remote conferencing. As will be discussed later, the manner of communications between the source and recipients may complicate information delivery depending on the type of network employed.

Multimedia applications are either interactive or streaming. Streaming applications are either unicast or multicast and are channel asymmetric: significant content flows in only one direction. Interactive applications tend to have content flowing, in part, in at least two directions although the flow may not be fully symmetric. Streaming applications usually do not require strict bounds on delay but are sensitive to delay jitter. Interactive applications usually require strict bounds on both or not at all, depending on the information content.

The information flow for multimedia applications is either continuous or intermittent. Applications with intermittent flow are not usually delay sensitive but tend to tolerate cell loss poorly. Examples include text files, still images, and graphics. For applications with continuous flow, such as video and audio, delay sensitivity depends on whether the application involved is interactive or streaming as mentioned above. Some cell loss is acceptable for continuous flows although the degree depends on the information source as well as the amount of compression involved.

B. PROBLEM SCENARIO

This section lays out scenario parameters for a tactical shipboard VTC application and discusses difficulties with preserving video quality using traditional video coders over heterogeneous networks.

1. Target Scenario

Using the IT-21 requirements as a baseline, the battlegroup tactical network is assumed to be a hybrid wireline/wireless ATM network. Shipboard networks employ an ATM backbone and provide complete ATM connectivity to the desktop, offering either

native ATM services or legacy LAN emulation over ATM. An ATM wireless network provides connectivity within the battlegroup. A centralized control station, usually the capital ship within the battlegroup, may manage access to the wireless network. This network is illustrated in Figure I.2.

Intrinsically, this arrangement offers asymmetric bandwidth depending on whether a connection remains shipboard or is ship-to-ship. Given the current capabilities of ATM network interface cards (NICs), workstations can expect a maximum bandwidth of 10-25 Mbps with correspondingly higher bandwidths across the backbone. However, given current technology, wireless data rates are far more limited. A reasonable assumption is a bandwidth of at least 1 Mbps, a value well within the capability of commercially available technologies, such as Multichannel Multipoint Distribution Service (MMDS) broadband wireless transmission. MMDS offers line-of-sight (LOS) service in the 2.1 GHz to 2.7 GHz band with data rates up to 1.5 Mbps. Satellite links complete the connectivity to land-based LANs but are not considered further here since their high latency precludes satisfactory performance for interactive multimedia.

The maximum quality of any multimedia application depends in part on available bandwidth (network services also play an equally important role). While the network described here provides for high bandwidth aboard individual units, networking between units is constrained by the wireless interface. Thus, deploying a tactical VTC application at the battlegroup level requires operating within this bandwidth constraint.

To provide a basis for the work presented here, a set of reasonable requirements for low-bit-rate tactical VTC is proposed below using international standards where possible to keep within the spirit of IT-21. Given the bandwidth constraints, both the audio and video streams must be compressed. Toll quality speech demands far less bandwidth than video and can be reasonably limited to 8 kbps or less using code excited linear prediction (CELP) speech coding [5]. Video bandwidth requirements depend on the desired resolution, frame rate, color depth, and the permissible tradeoff between compression gain and perceptual quality. Current low-bit-rate ITU multimedia standards, such as H.320 and H.324, use low resolutions and frame rates to enable acceptable video

quality [3]. Using these standards as a guideline, the tactical VTC transmits video signals at 10 fps using the Quarter Common Intermediate Format (QCIF) with a resolution of 176×144 pixels and targets bit rates in the range of 64-96 kbps. The primary color depth supported is 8-bit grayscale although 4:2:0 sub-sampled 24-bit color [6] is a possible option. These requirements are summarized in Table I.1.

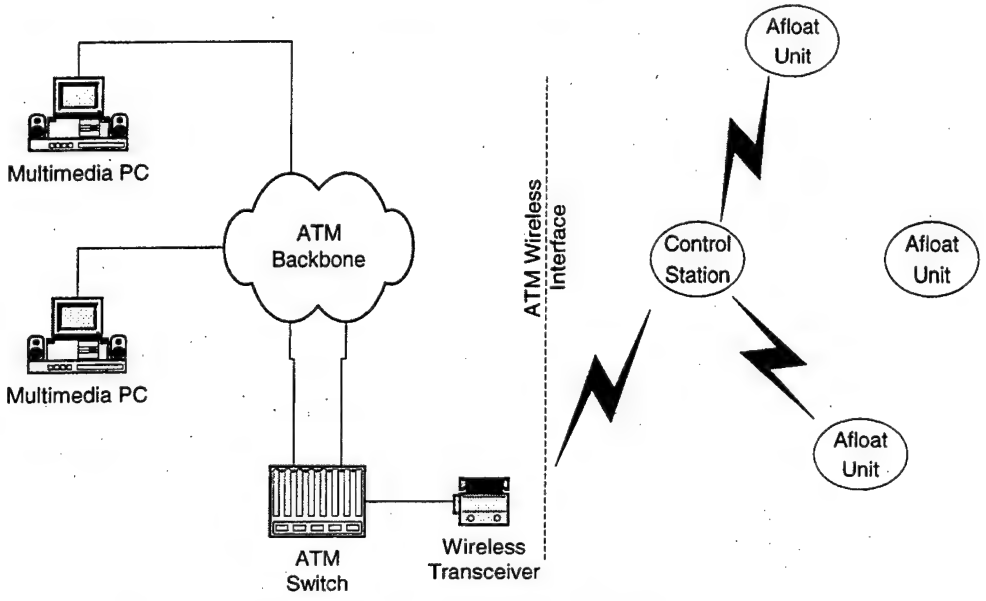

Figure I.2: Hybrid ATM Wireline/Wireless Network.

VTC Stream	Parameter	Value
Video	Bandwidth	64-96 kbps
	Resolution	176×144 (QCIF)
	Frame Rate	10 fps
	Color Depth	8-bit gray/4:2:0 24-bit color
Audio	Bandwidth	< 8 kbps

Table I.1: Tactical VTC Multimedia Requirements.

2. Video Compression and Robustness

Given the parameters in Table I.1, a video compression gain of approximately 31 to 1 is required to transmit 8-bit grayscale, assuming an average available bit-rate of 64 kbps. Such gains are easily within the capability of current video coding standards, such

as H.263 and MPEG-1/2. However, traditional video compression schemes are not particularly suitable for multicast transmission over packet-based networks.

Video codecs compress the original video stream by removing the least perceptually relevant content and by encoding only the differences between successive frames caused by motion. Unfortunately, packet-based networks invariably drop packets due to congestion, even in network architectures offering QoS guarantees, such as ATM networks. Due to the high compression gains required for transmission, each packet contains a significant amount of information. The loss of a single packet corrupts a portion of a frame or an entire frame depending on the decoder's ability to resynchronize with the incoming bit stream [7]. With motion compensation, any visual error artifacts introduced may persist for many frames past the initial point of corruption (until the next I-frame in an MPEG stream and possibly indefinitely in H.263 [8]). The effect of packet losses grows more significant as bit rate decreases.

The problem of packet losses may be mitigated within the network or at the application layer. Within the network, appropriate QoS guarantees can reduce cell losses to a level such that any quality degradation due to transmission errors is acceptable. However, the required cell loss rates can be quite small, on the order of 10^{-6}, which requires a large allocation of bandwidth to achieve. Two common approaches to improving error robustness at the application level are to use codecs without motion compensation, such as Motion-JPEG [9], or to vary the bit rate in response to the estimated degree of congestion within the network. Motion-JPEG compresses each frame individually, thereby greatly improving robustness since visual artifacts are confined to the affected frame. However, robustness comes with lower compression gains, and, therefore, Motion-JPEG delivers unacceptable quality at low bit rates. If the source coder is *controllable* [11], network feedback reports can be used to modify the demand placed on the network by changing the quality of video transmission. While this approach provides no inherent improvement in the error resilience of the video stream, but it does try to mitigate the effects of congestion on the received video stream.

However, designing a scheme for controlling the source rate is difficult when multicast transmission over a heterogeneous network is considered. A heterogeneous network may be defined as one in which end-users are stratified by available bandwidths and processing and display capabilities [12]. Using feedback to monitor congestion within the network and then making appropriate changes to the outgoing video stream becomes problematic as multicast group size increases or as the network topology grows more complex. Feedback messages may potentially add to congestion depending on the periodicity of transmission. More importantly, since each user represents a different path through the network, each connection potentially experiences a different level of congestion. The controllable application is faced with a quandary in responding fairly if only a small number of members within the multicast group are experiencing congestion. Stratification poses a further problem during transmission of real-time video since each user has different expectations and tolerances with regard to video quality. Users with high bandwidth expect high quality video while users with low bandwidth are generally satisfied with less. Meeting the varied expectations with a single video stream is clearly impractical and transmitting multiple video streams with gradations in quality requires greater bandwidth.

C. DISSERTATION OBJECTIVES

Given the interest in deploying VTC applications over tactical networks such as those envisioned by US Navy's IT-21 initiative, distributing the video stream while maintaining acceptable quality involves reconciling the requirements of multimedia applications with the capabilities of tactical networks. As discussed in the previous section, video is bandwidth intensive and highly sensitive to transmission errors. A tactical network may be characterized as low bit rate, unreliable, and heterogeneous. Solving the distribution problem solely in terms of coder design or network design is less effective than developing a unified solution that reaches across the application-network boundary.

Accordingly, this dissertation investigates issues related to distributing low-bit-rate video within the context of a teleconferencing application deployed over a tactical ATM network. The main objective is to develop mechanisms that support transmission of low-bit-rate video streams as a series of scalable layers that progressively improve quality. These mechanisms exploit the hierarchical nature of the layered video stream along the transmission path from the sender to the recipients to facilitate transmission. Specifically, the approach proposed in this dissertation works across the application-network interface by coding the video stream into layers, shaping the resulting layered video stream prior to entry into the network, and prioritizing service in accordance with the relative perceptual importance of each layer. The resulting distribution path is illustrated in Figure I.3.

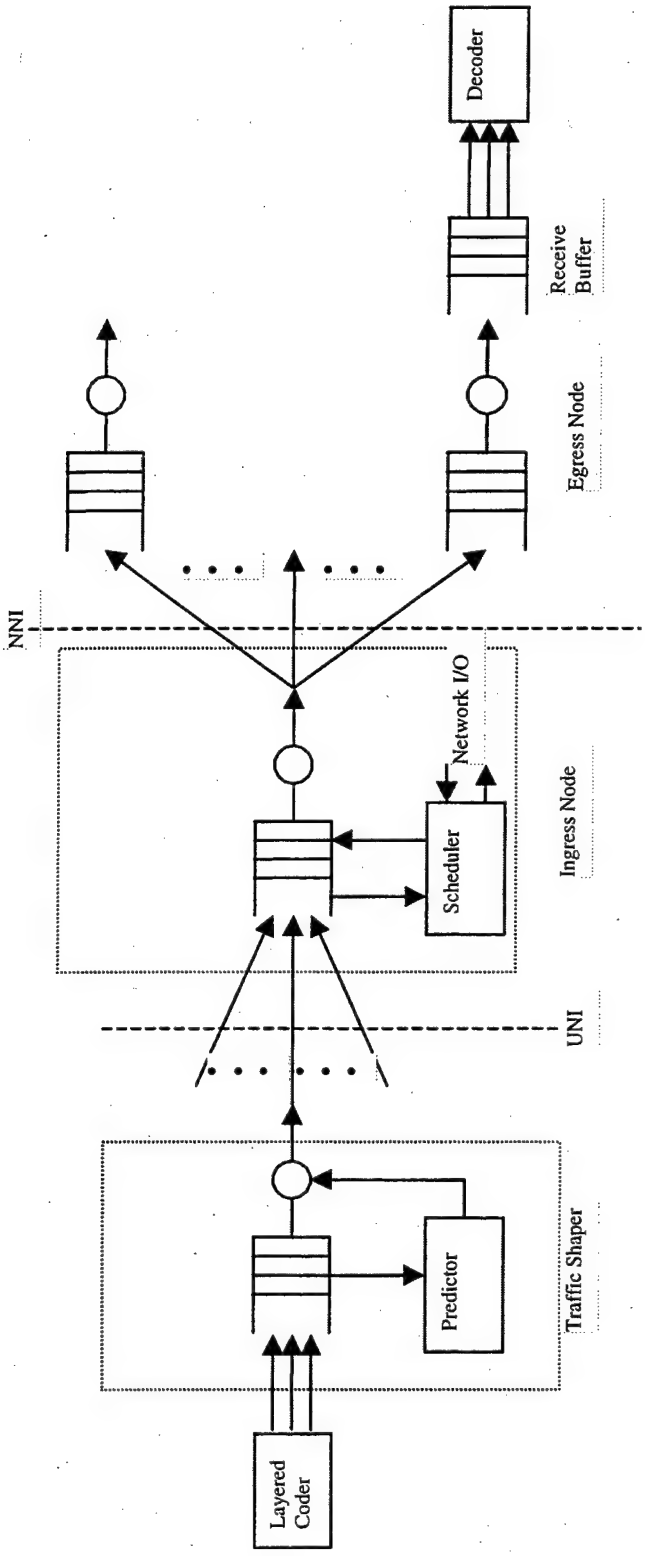

Figure I.3: Functional Diagram for a Multicast VTC Application.

Each of these mechanisms centers on dividing the video stream into an independently decodable base layer that guarantees a minimum, acceptable level of quality and several enhancement layers that increase quality in a hierarchical manner. Transmitting video in layers has several inherent benefits. The layered structure provides a means for implementing open-loop congestion control by allowing recipients to drop layers exhibiting high packet loss rates, thereby reducing network loading [12]. Earlier work by Rhee and Gibson [13] indicates that layered video exhibits improved resilience to bit errors introduced during transmission since spreading bit errors across multiple layers has less impact on the reconstructed video.

Here, a new layered coder design tailored to video teleconferencing in the tactical environment is proposed. Specifically, the coder is optimized for VTC video scenes consisting of low motion video, such as a "talking head," and static scenes corresponding to presentation slides. The concession to the tactical environment is an emphasis on low-bit-rate coding, low-complexity coding for low delay and power requirements, and inherent robustness to minimize the effect of packet losses and bit errors. Two major problems are considered. The first is the notion of how to *effectively* map frequency content to the requisite number of layers and thus creating the required perceptual hierarchy. A generalized layering scheme presented uses the fast Haar transform to segregate frequency content into subbands; these subbands are then grouped by perceptual relevance to form the required number of layers. However, a layering scheme suitable for low-motion video is unsuitable for static slides. Static slides place a much greater emphasis on high-frequency content, and an appropriate layering scheme is included with the coder design. The coder adapts to the current video type by shifting to the correct layering structure.

The second problem is developing a rate control scheme for the layered video coder. Rate control is a requisite for maintaining a desired QoS level in an ATM network, but the use of multiple quantizers complicates developing an optimal rate controller appropriate for a real-time application. A suboptimal rate control mechanism that reduces the k-dimensional rate-distortion problem resulting from the use of multiple

quantizers to a 1-dimensional problem by creating a single rate-distortion curve for the coder in terms of a suboptimal set of *k*-dimensional quantizer vectors provides a more appropriate alternative. Rate control can thus be simplified into a table lookup of a codebook containing the suboptimal quantizer vectors.

The manner in which the compressed bit stream is transmitted to the network has a profound effect on queuing efficiency and therefore the bandwidth required to meet the required QoS. Smoothing the video traffic reduces variation and uncertainty in the arrival process and improves queuing efficiency. Here, a traffic shaper is employed to deterministically smooth the entire stream, all layers included, to maximize queuing efficiency. The only drawback to smoothing is the insertion of additional delay in the transmission path due to the need to buffer an entire encoded frame prior to transmission. However, a new scheme is proposed that partially offsets the delay. The traffic shaper is also responsible for interleaving cells from each layer for transmission within the outgoing stream. Order of arrival into the queue appears to affect scheduling performance in priority-based scheduling systems [16].

Layered video traffic offers another dimension to the scheduling problem as well as an avenue for reducing the impact of network congestion on the overall quality of the reconstructed video. Since video is transmitted as a base layer and a series of enhancement layers, a hierarchical priority system is appropriate. During periods of no congestion, the layered video connection is serviced at its required QoS without regard to the layering structure. During network congestion, emphasis is placed on servicing the most perceptually important layers, starting with the base layer, and denying service to the least important layers. Conceptually, the overall connection is granted a certain bandwidth. As cell loss increases due to congestion, the bandwidth is reallocated to support only the most important layers.

However, the impact of an individual cell loss may not be viewed in isolation. Another factor to consider is the temporal dependence between adjacent cells in ultimately reconstructing the video sequence. Both cell losses and bit errors in transmission create gaps in the incoming bitstream causing the decoder to lose the

synchronization required to recognize codewords within the stream. The decoder then must parse forward within the bit stream until a marker is found to re-enable synchronization. Therefore, if a cell is dropped from the queue, all cells up to but not including the cell containing the next marker are not useable and will not be decoded. This situation can be exploited by reacting to cell loss by searching for related cells rendered unusable and discarding them to open scheduling opportunities for other cells.

D. DISSERTATION ORGANIZATION

The dissertation is organized as follows. We start with a discussion of general multimedia network architectures and traditional video codec designs. Next, the elements for improving network distribution of low-bit-rate video are presented. These elements include design of a suitable low-complexity layered video coder for tactical environments, a traffic-shaping scheme to maximize queuing and scheduling efficiency, and network scheduling algorithms that provide QoS support for layered video while maximizing perceptual quality during periods of congestion.

Chapter II begins with an overview of transmission of multimedia traffic in both the IP and the ATM environments. ATM and a brief discussion of related ITU standards for multimedia terminals are covered. Since layered video follows a strict hierarchy in regard to perceptual importance, identifying layers within the network is crucial to implementing priority-based scheduling. Also, as dropped cells may corrupt future portions of the video stream, either within a layer or across all layers, identifying logical resynchronization points with the stream allows the scheduler to make intelligent decisions on when to discard cells. Accomplishing each of these tasks is dependent on the manner in which the layered video stream is transmitted within an ATM network. Therefore, two approaches are examined: multiplexing all layers over a single virtual channel or assigning individual layers to separate virtual channels.

Chapter III provides an overview of hybrid video coding along with a brief introduction to the three components of video coding: transforms, quantization, and entropy encoding. The notion of wavelet-based image compression is presented as a

motivation for layered video transmission. Chapter IV examines the problem of layered coding for both low-activity motion video and static presentation slides. A heuristic approach to designing layering schemes for motion video is presented and a particular scheme for low-bit-rate video is proposed. As a layering structure for motion video is unsuited for static presentation slides, another layering structure is proposed emphasizing the greater perceptual importance of the high frequency content. The problem of rate control for layered coding is examined and a simple open-loop controller is proposed.

Chapter V discusses the concept of traffic smoothing for increasing queuing efficiency and scheduler performance. An integrated smoothing scheme is proposed that smoothes traffic at three time scales: interframe, intraframe, and across the layer hierarchy. Implementation within the context of an ATM network is also considered. Chapter VI addresses the issue of scheduling layered video traffic. Several algorithms are proposed to maximize throughput while exploiting the opportunities provided by layered video to reallocate bandwidth within a connection as required to preserve the higher priority layers. A cell-discard policy is also discussed that represents the interdependence of cells in the traffic flow, both within a layer and across layers. Simulation results illustrating the different algorithms are presented and discussed.

Chapter VII summarizes the significant contributions made in the dissertation and provides concluding remarks along with a discussion of possible topics for future research in layered video transmission and related areas.

Appendix A presents the OPNET process models used to validate the behavior of the layered scheduling algorithms presented in Chapter VI. Appendix B presents a suitable video traffic model used to simulate the behavior of a rate-controlled video traffic stream.

II. NETWORK ARCHITECTURES FOR MULTIMEDIA TRAFFIC

Before introducing the topics of video compression and scheduling, we examine integrated services network architectures appropriate for video teleconferencing. We start by considering the characteristics of a generic *m* to *n* VTC application. VTC applications are inherently real-time interactive, transmit continuous media as well as discrete, and operate in multicast mode. The interactive and continuous nature of the application suggests that strict bounds are required on both delay and delay jitter. Since both video and audio traffic are generally compressed, packet losses must be limited to avoid excessive reconstruction errors. Summarizing, the characteristics of VTC applications imply the following requirements: multicast support, QoS guarantees, and real-time support. Based on these requirements, two network architectures provide a suitable basis for VTC [3]: IP-based networking in conjunction with RTP and ATM networking.

The purpose of this chapter is to refine the networking scenario underlying the VTC application and provide a context for the work presented in this dissertation. While multicast IP is briefly considered, a wireless ATM network appears more suitable for tactical VTC applications and is covered in far greater detail. Emphasis is placed on describing ATM's support for different traffic types, QoS support, and connection setup using a simple layered protocol model to indicate where each level of functionality is implemented. The ATM cell format is examined, and an overview of ATM multicast implementations is presented. Two other related topics are covered in some detail: a brief introduction to wireless networking, focusing on the data link control and physical layers, and coverage of ITU multimedia terminal standards that pertain to ATM networks.

The final issue considered is support of layered video traffic within the context of established ATM networking standards. The first problem is how to map individual video layers onto ATM connections. All of the layers may be interleaved over a single

logical connection or transmitted separately using individual connections. The implications of both approaches are considered and presented along with the attending advantages and liabilities. The second problem is facilitating layer identification within the network to implement an appropriate scheduling algorithm. In some cases, it is also valuable to identify other elements within each layer, such as the positioning of frame and group of blocks (GOB) headers. Identification is complicated by the deliberate simplicity of the ATM cell header since the user has limited means for altering fields within the header. Two cell tagging schemes are presented to accommodate this, one for the single connection case and the other for the multiple connection case.

A. LEGACY IP-BASED NETWORK

Although the TCP/IP protocol suite is the dominant commercial architecture for internetworking, TCP/IP is not practical for real-time, multimedia applications. Still IP-based networks are so prevalent that incentives exist for working within the limitations imposed by IP to add some support for real-time traffic. The current approach is to use RTP over UDP/IP to provide real-time support for a video application as illustrated by the protocol stack illustrated in Figure II.1. The following paragraphs consider both TCP/IP networking and RTP over IP; the latter is termed the legacy approach to real-time networking. TCP/IP is considered primarily to show how the design decisions, while appropriate for the type of traffic originally envisioned, preclude real-time support. Discussion of the lower layers is deferred until later when wireless networking is considered.

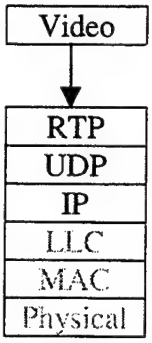

Figure II.1: IP-Based Network Protocol Stack for Real-time Traffic.

1. IP and Multicast IP

In regard to real-time traffic, IP is effectively neutral. IP provides a connectionless service to higher layers, providing only "best effort" delivery of datagrams [19]. Best-effort service does not guarantee that any data transmitted will ultimately be delivered or arrive in any particular order. Connectionless service was chosen for IP since datagrams traveling through different networks might encounter a variety of protocols. By offering only an unreliable service, IP requires very few services from the constituent networks traversed by datagrams. Any additional end-to-end services, such as a reliable, connection-oriented service, are added by transport layer protocols, such as TCP, if needed. However, best-effort service precludes any notion of QoS by definition. Although higher layers may add additional functionality to control information loss, other QoS parameters, such as delay and delay jitter, cannot be guaranteed. Even worse, if any part of a network transmission path includes an IP network, no explicit QoS guarantees are possible regardless of the capabilities of the other networks in the path.

IPv4 has been extended through various efforts to provide multicast functionality though support must be regarded as experimental since currently most IP routers do not explicitly provide multicast service. The best example of multicast IP is MBone (multicast backbone), an outgrowth of early multicast experiments during the formulation of the IP multicast protocol [20]. Mbone consists of a virtual network of multicast routers or *mrouters*. Multicast packets are transmitted point-to-point between mrouters, using tunneling as necessary to traverse ordinary routers [21]. Several audio and video

tools have been written to take advantage of Mbone, but they are restricted primarily to the Unix platform [1]. The next generation of IP, IPv6, explicitly supports multicast functionality [18].

2. Transport Layer Protocols

TCP is a transport protocol that provides the reliable, connection-oriented service lacking in IP and guarantees sequential delivery of data to the application layer. However, this very service precludes the use of TCP for real-time multicast applications [18]. TCP is a point-to-point protocol; TCP connections are established between two end users. Reliability and sequencing are provided through a system of acknowledgments and retransmissions [19]. However, real-time applications have stringent delay requirements and retransmitted segments usually cannot arrive in time to provide a benefit. In this case, retransmissions merely waste bandwidth. TCP also includes a window-based flow control scheme to prevent faster systems from overwhelming slower systems with data and to implement congestion control schemes. However, the same scheme impedes delivery of streaming data.

For these reasons, UDP is favored for real-time traffic, offering simple transport layer access to IP with low overhead. While UDP provides no explicit support for real-time applications, real-time traffic is not impeded as in the TCP case [18].

3. Real-time Transport Protocol

RTP is a lightweight transport protocol for real-time applications and employs UDP for access to both IP and multicast IP. RTP does not provide either reliable service or QoS guarantees since the underlying IP layer precludes these services. RTP does provide a framework of services to the application that allows the application to monitor and compensate for the actual QoS the network is *delivering* to the recipients. RTP follows the concept of application-level framing [11] as posed by the following scenario. The sending application transmits data continuously to one or more receiving applications. Each receiving application is able to accept less than perfect delivery and still continue operating, thus negating the need for retransmissions. For example, a video

decoder parses past missing data and resynchronizes as required to restart decoding. However, each receiver does monitor the QoS provided by the network, in terms of delay, delay jitter, and packet loss, and relays the information back to the sender. Taken collectively, the feedback reports indicate network conditions and provide an opportunity for the sender to adapt in hopes of obtaining better QoS. If receivers report high packet losses, indicating possible network congestion, the sender might move to a lower-quality transmission to place a smaller demand on the network. To benefit from RTP, the application must be *controllable*, that is, able to adjust bandwidth requirements dynamically as dictated by network conditions. A video coder, for example, could reduce frame rate, resolution, or perceptual quality [9].

RFC 1889 specifies both a data transfer protocol, simply termed RTP, and a RTP control protocol, RTCP [10]. RTP supports either unicast or multicast transmission by organizing participating RTP entities into a session. Each entity transmits data to the session through a single UDP port using an application-level packet format defined by the protocol. RTP packet headers identify the payload type: the media type (audio or video) and the format (G.728 audio or H.261 video) [22]. The header also provides a source identifier to indicate the multicast group generating the data, a sequence number for loss detection, and a timestamp for recording the time the first byte of data was generated. The timestamp allows synchronization among different streams.

RTCP provides for feedback reports to sending applications as well as reports to all members of the multicast session [10][18]. Reports are transmitted through a separate UDP port from RTP packets. Receiver reports provide feedback on observed QoS to the sending entity. Sender reports are used to alert participants when multiple source identifiers are related, such as synchronized audio and video streams, and should be received together. Each session member also periodically sends status reports that collectively allow other members to estimate the size of the session. Session size is used to scale the report transmission rate to avoid overburdening the network.

An important point is that RTP does not provide a mechanism or algorithm for determining the manner in which the sender interprets feedback reports and adjusts

network demands. Instead the application must be written to take advantage of RTP, which suggests that RTP should be viewed as more of an application framework than a complete networking protocol [18].

4. Suitability of RTP/IP for VTC

The introduction to this section indicated three features required for a networking architecture to fully support video conferencing. The legacy RTP/IP network architecture provides adequate real-time and multicast support, yet the architecture falls short in two areas. First, applications use RTCP receiver reports to mitigate the effects of congestion. With large or heterogeneous networks, the reports may vary significantly since each receiver experiences different network conditions. This greatly complicates the control issue although it is correctable to some extent with RLM [45]. Second, and more significant, the lack of QoS guarantees may lead to unsatisfactory reconstruction of the audio and video streams. IP routers do not guarantee QoS since IP routing does not incorporate the concept of resource reservation and only provides service through variants of first-come, first-serve (FCFS) scheduling. The new Resource reSerVation Protocol (RSVP) has been developed to provide support for QoS under the proposed Integrated Services Architecture (ISA) [18]. Each router running RSVP must implement an admission control scheme, a scheduling scheme, and be able to classify packets according to QoS requirements. At this point, RSVP is not widely implemented and its capabilities are already duplicated by the more mature ATM network architecture.

B. ATM NETWORKS

ATM grew out of the desire to utilize the high bandwidth available from optical fiber to create a Broadband Integrated Services Digital Network (B-ISDN) that is able to support audio, video, and data services within the same network [27]. In contrast to TCP/IP, where the end-user transport layers provide only reliable service and network delivery is best effort, ATM networks provide QoS guarantees. ATM guarantees QoS by comparing the caller's QoS requirements to available network resources and then allowing a connection if sufficient resources exist [18]. Resources are reserved for the

duration of the connection. ATM distinguishes among several different service or traffic classes, such as the real-time and non-real-time traffic at constant and variable bit rates, and provides support through a combination of QoS primitives and transport layer adaptation.

ATM represents a medium between PSTN circuit-switched networks and connectionless packet-switched networks. ATM uses virtual circuits to simplify switching decisions but allows several connections to be multiplexed over a single physical interface to promote efficient bandwidth utilization. Virtual circuits imply connection-oriented service, but ATM also provides the equivalent of connectionless service to support the widest range of applications possible.

ATM was designed to support high bit rate connections, such as OC-3 (155 Mbps) and OC-12 (622 Mbps) over fiber [23][24]. The decision to employ fiber, a physical medium with extremely small bit-error-rates (BER), allows ATM to minimize both error and flow control functionality. Minimizing these capabilities reduces overhead in processing ATM cells and decreases the header bits required per cell, thus allowing fast switching speeds and efficient data transport. High speed switching is further supported by use of small, fixed-length cells.

1. ATM Protocol Model

The ATM protocol model is shown in Figure II.2 [23]. The protocol model consists of three separate planes: management, control, and user. The management plane provides management functions and exchanges information between the control and user planes. The control plane deals with call establishment, connection control, and call release. To provide these functions, the control plane has access to the network and separate signaling protocols and cell definitions. The user plane supports transfer of user information by providing such functionality as flow and error control, timestamps for synchronization, and sequencing.

The user plane includes the ATM Adaptation Layer (AAL), the ATM layer, and the physical layer. The AAL is a service dependent layer and adapts information streams from higher layers for transmission over ATM. Example streams include compressed

video, constant bit rate (CBR) audio, or even IP datagrams. Each has distinct service requirements. The AAL maps data and service requirements from these streams to services provided by the ATM layer. The ATM layer provides data transport using cells over an end-to-end logical connection and controls access to the underlying physical layer.

The physical layer is medium dependent. The physical layer includes two sublayers: physical-media *dependent* (PMD) sublayer and transmission-convergence (TC) *independent* sublayer [23]. The former deals with aspects that are dependent on the transmission medium selected (e.g., bit timing and line coding). The latter handles issues that are independent of the transmission medium characteristics, such as error control or determination of cell boundaries in the physical layer payload. ATM specifies SONET, a fiber standard that provides synchronous time-multiplied transmission at high bit-rates, as the basic physical layer interface. Other physical layer interfaces, such as UTP [25][26], are specified to promote interoperability.

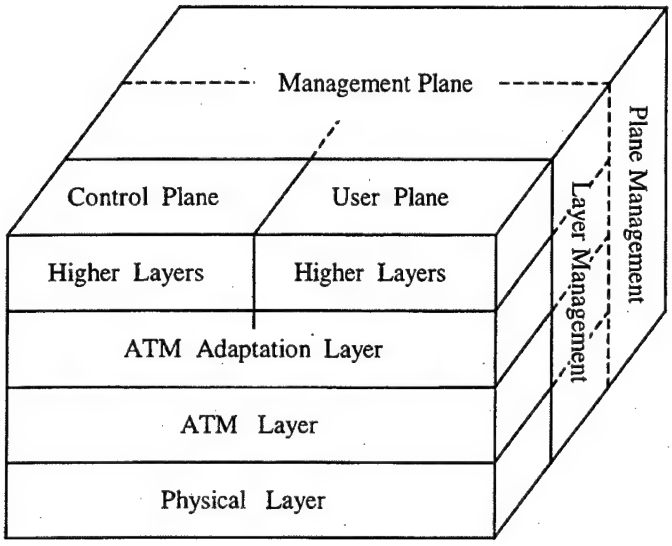

Figure II.2: ATM Protocol Architecture [23].

2. Logical Connections

End-to-end connections in ATM are defined in terms of a virtual channel connection (VCC) and a virtual path connection (VPC). Figure II.3 illustrates the role of VCCs and VPCs within an ATM network. VCCs are created dynamically between two end users to provide a unidirectional channel for ATM cells carrying user data and are terminated at call release. Cells are carried in sequence. VCCs are also set up between an end user and the network to carry control signals and between network nodes to facilitate network management and routing. These connections cross the user-network interface (UNI) and the network-network interface (NNI), respectively.

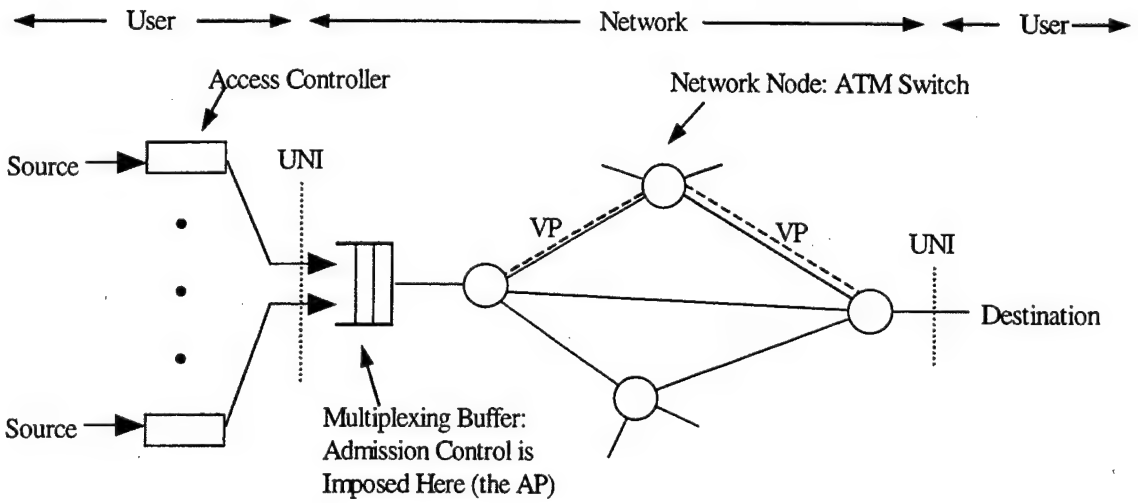

Figure II.3: ATM Network Configuration [27].

ATM networks include a higher level of connectivity in the form of virtual paths. The virtual path concept is motivated by the trend toward increasing bandwidth, which also increases the possible number of connections a channel may carry. Compared to IP networks, ATM's circuit-oriented structure and QoS guarantees incur greater control costs. Since these control costs scale with the number of connections, virtual paths decrease cost by reducing the number of connections managed by the network. A VPC represents a network-defined, end-to-end connection representing a set route through the

network and providing a specified QoS such as bandwidth. Each VPC carries multiple VCCs with these same end-points, and all associated cells are switched along the same path. Since most of the work required to establish a connection (reserving capacity and calculating routes) is performed when a VPC is established, call setup time for new VCCs is greatly reduced.

3. ATM Cell Format

ATM employs fixed-size cells consisting of a 5-octet header and a 48-octet information field. The cell header format differs depending on whether the cell is entering the network (UNI) or moving within the network (NNI). Figure II.4 shows the ATM cell format at the UNI. NNI ATM cells do not retain the generic flow control (GFC) field; instead they use the bits to expand the virtual path identifier (VPI) from 8 to 12 bits.

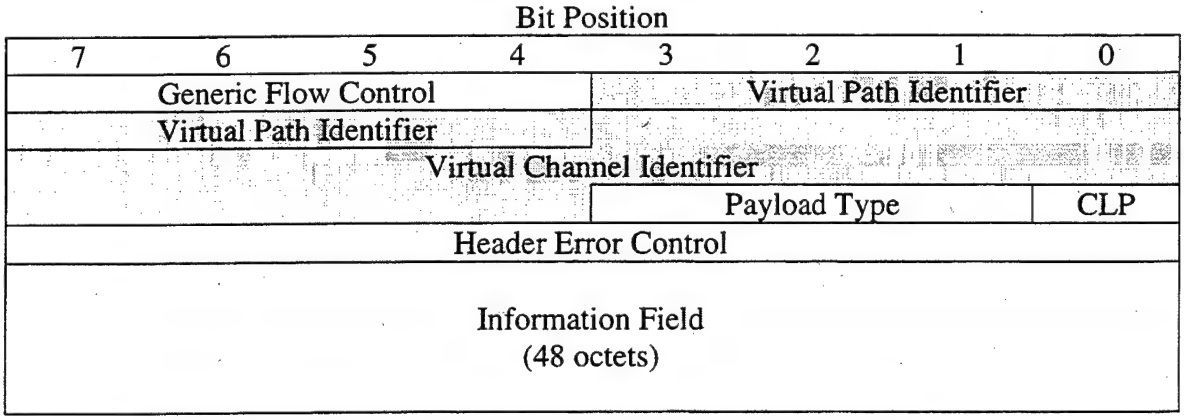

Figure II.4: ATM Cell Format at the UNI [23].

The GFC field is used to control cell flow at the UNI although application remains an area of active study [18]. The GFC is not carried end-to-end and is overwritten by ATM switches to expand the VPI.

The VPI identifies a routing path within the network. The field width is 8 bits at the UNI and 12 bits within the NNI, thereby allowing a greater number of virtual paths within the network. The virtual channel identifier (VCI) identifies an end-to-end routing path and functions similar to the ports in TCP or UDP.

The payload type (PT) is a 3-bit field used to indicate the type of data in the information field. A high order bit of 0 indicates a user data cell; 1 indicates either a resource management (RM) cell or a cell carrying maintenance information. The second bit is initially cleared at the UNI. Within the NNI, a switch sets the second bit whenever congestion is experienced. Switches downstream can monitor this bit to guage network conditions. The third bit is the service data unit (SDU) type bit and allows the user to designate two types of SDUs. One use of the SDU bit is to implement different service strategies for ATM cells based on their content.

The cell loss priority (CLP) field is set by the user to indicate the relative priority of cells in case congestion forces a switch to discard cells. A value of 0 indicates higher priority, and the cell should be dropped only as a last resort; 1 indicates a lower priority cell that a switch may drop to ease congestion. As part of call setup, the user negotiates a contract with the network and agrees to transmit data in accordance with various traffic parameters. The user may negotiate separate contracts for CLP = 0 and CLP = 1 traffic. Network switches also set the CLP bit for any data cell in violation of its traffic contract even if the switch has sufficient capacity to transmit the cell. Subsequent switches may then discard the cell as required.

ATM cells include an 8-bit header error control (HEC) field calculated based on the first four octets of the header. The HEC allows detection of errors and correction of single-bit errors. If a multi-bit error is detected, the cell is discarded. No error detection is provided for the information field.

4. ATM Service Classes

ATM is designed to support a wide range of applications: from interactive applications, such as video and multimedia conferencing, to distribution services, such as archive retrieval and document browsing [27]. Recall that each application transmits a sequence of cells through a virtual channel connection. Providing the desired QoS to a new VCC depends on the new connection's traffic flow characteristics as well as the characteristics of existing VCCs. Traffic handling, from call acceptance to network scheduling, is therefore simplified by defining discrete service categories. The ATM

Forum has defined five ATM layer service classes as shown in Table II.1 [28]. Each VCC established receives service in accordance with one of these categories.

Interactivity	Service Class
Real-time service	Constant bit rate (CBR)
	Real-time variable bit rate (rt-VBR)
Non-real-time service	Non-real-time variable bit rate (nrt-VBR)
	Available bit rate (ABR)
	Unspecified bit rate (UBR)

Table II.1: ATM Service Classes [28].

Real-time services are characterized by low tolerance for delay and delay jitter. Applications that involve human interactivity, such as video conferencing, are real-time since excessive delay degrades the perception of true interactivity and jitter impedes the smooth playback of audio and video. The two services defined for real-time service, CBR and rt-VBR, are distinguished by variation in data rate. CBR, as expected, transmits data at a fixed rate and is the easiest service to support. Applications include both compressed and uncompressed data. Toll-quality PCM speech requires a constant data rate of 64 kbps. H.261 was designed to support transmission over one or more ISDN B channels and compresses video at a multiple of 64 kbps. CBR is commonly employed for uncompressed applications, such as broadcast quality video conferencing and interactive audio. Rt-VBR applications have data rates that are "bursty" and time-varying and are characterized by a mean bit rate and a peak bit rate. Compressed video is inherently VBR since compression gain naturally varies with each frame depending on scene content (see Chapter III). Rt-VBR is more difficult for networks to support but provides greater flexibility than CBR. VBR streams may be statistically multiplexed over the same channel for more efficient use of bandwidth.

Non-real-time services are intended for bursty traffic without stringent requirements on delay and jitter, thus giving a network more flexibility in dealing with these traffic flows. Nrt-VBR applications generate VBR data that does not require strict

limits on delay but does require some upper bound. Examples include banking and airline transactions [18]. UBR service is best effort service similar to that provided by IP-based networks. UBR connections receive no dedicated resources; bandwidth is provided dynamically from spare channel capacity not utilized by CBR and VBR traffic. ABR improves upon UBR's best effort service. ABR applications specify both a minimum cell rate (MCR) and a peak cell rate (PCR). At any time, the network ensures a fair allocation of resources among all ABR connections such that each connection receives at least their MCR, and possibly up to the PCR, depending on available capacity. TCP connections and LAN traffic commonly employ ABR service. Figure II.5 shows how channel capacity could be allocated to each service category.

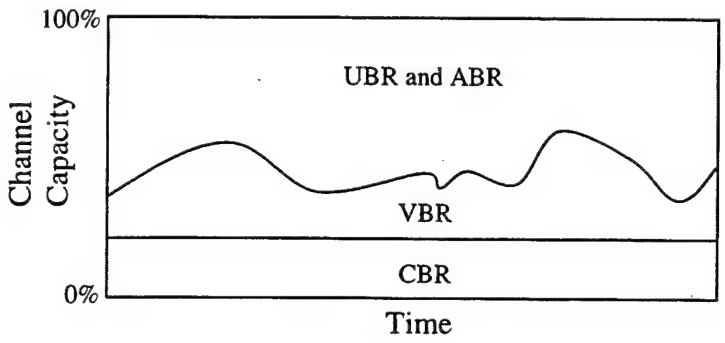

Figure II.5: Bandwidth Allocation for ATM Service Categories [18].

At call setup, a user requests service by supplying the network with traffic descriptors that characterize the cell flow and the required QoS. The exact parameters provided are service dependent. Traffic descriptors allow the network to determine if sufficient resources are available to support the connection's QoS requirements. For example, a user requesting rt-VBR service must supply the PCR, the sustainable cell rate (SCR), and the maximum burst size of cells (MBS). A CBR connection provides only the PCR. The QoS desired is specified in terms of cell delay variation (CDV), maximum cell transfer delay (maxCTD), and cell loss ratio (CLR). Real-time services require all three QoS parameters be specified. Non-real-time services do not specify any QoS parameters except for nrt-VBR, which specifies CLR.

A connection is accepted only if network can reserve sufficient resources while maintaining the QoS of existing connections. Assuming the connection is accepted, the traffic descriptors and QoS parameters form a traffic contract between the user and network. The user agrees to transmit in accordance with the traffic parameters. In turn, the network guarantees the QoS parameters for the duration of the connection. Once, the connection is active, the network performs traffic policing to ensure compliance. If the user violates the traffic contract, perhaps by exceeding the SCR, offending cells may be tagged using the CLP bit or discarded.

5. ATM Adaptation Layer (AAL)

Referring back to Figure II.2, the AAL provides services to applications or other transfer protocols not found in the ATM layer. To minimize the number of AAL protocols required, ITU-T Recommendation I.121 defines four generic service classes[1], A-D, based on three application service requirements [29]: bit rate (constant or variable), the timing relationship between the source and receiver (required or not), and the connection mode (connectionless or connection-oriented). These service classes are more general than the previously described ATM layer service classes and do not include either formal traffic descriptors or QoS parameters. In addition to these application service requirements, ITU-T Recommendation I.362[2] provides example services that the AAL may provide to enhance the ATM layer including [30]: handling transmission errors, segmentation and reassembly to map user data to the 48-octet information field in ATM cells, handling lost and misinserted cells, and flow and timing control.

To distinguish between data handling and service dependent functionality, the AAL is divided into two sublayers. The convergence sublayer (CS) provides service-dependent functions and a service access point (SAP) for applications. Functionality within the CS is further differentiated into the service specific CS (SSCS) and the common part CS (CPCS). Discussion here focuses on the CS as a composite entity. The

[1] Two other classes, X and Y, considered for a raw cell delivery service have been dropped.

[2] I.362 has been superceded by the ITU-T F.600 and F.700 Series recommendations.

SSCS and CPCS are individually addressed only when required. The segmentation and reassembly (SAR) sublayer segments user data to fit within the 48-octet length of the ATM cell information field and reassembles user data correctly at the destination.

Segmentation is shown in Figure II.6. The higher layer delivers a protocol data unit to the CS sublayer. The CS sublayer adds either a header or a trailer or both and pads the CS-PDU as required. The SAR breaks up the CS-PDU, optionally adds a header and/or a trailer to each segment such that the resulting SAR-PDU is 48 octets in length. The SAR-PDU then fits within a single ATM cell for transmission. At the receiver, each of these steps is simply reversed.

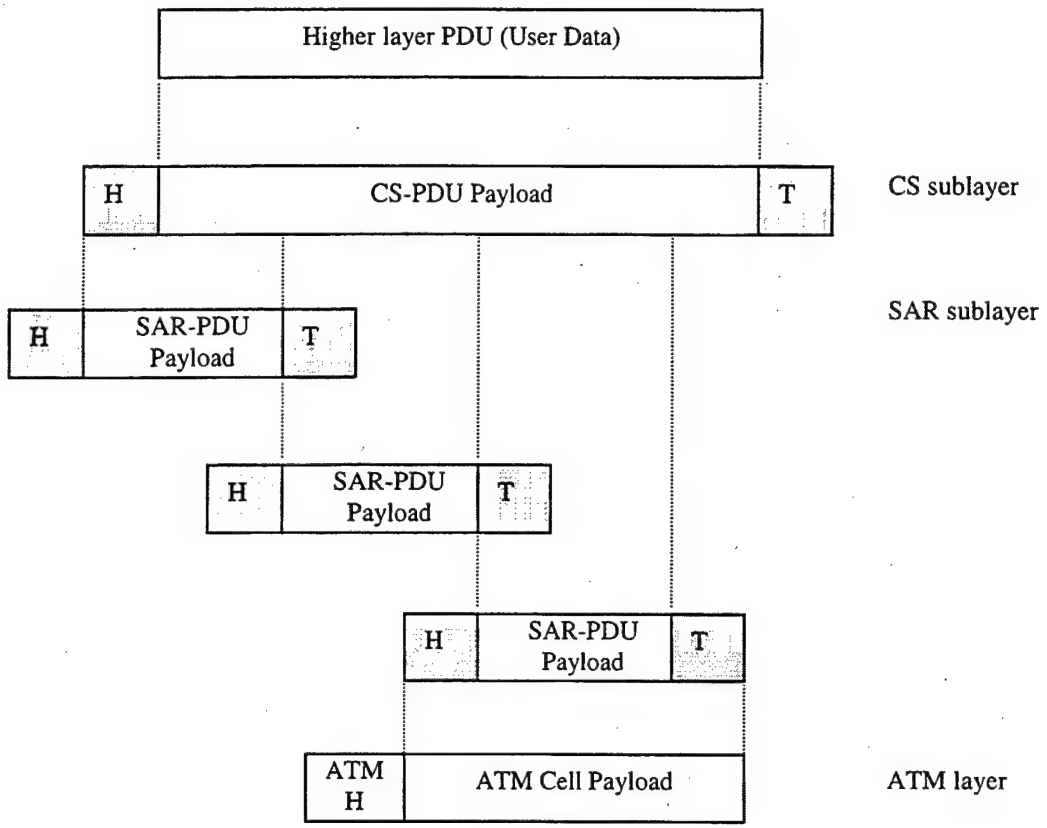

Figure II.6: Segmentation at the AAL [18].

The ITU-T originally proposed five AAL protocols [31], Types 1 to 5, but later combined Types 3 and 4. The relationship between the generic service classes proposed

by I.161 and the AAL protocols is shown in Table II.2; the protocols do not necessarily map to individual service classes.

	Class A	Class B	Class C	Class D
Timing Relation Required	Required		Not Required	
Bit Rate	Constant	Variable		
Connection Mode	Connection Oriented			Connectionless
AAL Protocol	Type 1	Type 2	Type 3/4	
			Type 5	

Table II.2: AAL Protocol Mapping to Service Classes [18].

The AAL protocols in Table II.2 map in an interesting manner to the ATM layer service classes shown in Table II.1. The most widely used protocols are AAL1 and AAL5. AAL1 is for connection oriented CBR traffic, matching the ATM layer CBR service. AAL5 is also connection oriented but supports VBR traffic. AAL5 assumes higher layers perform connection management and that the ATM layer produces minimal errors. As a result, AAL5 has low processing and transmission overhead and adapts well to existing transport protocols, such as TCP. These features make AAL5 the most versatile AAL protocol, and AAL5 is used with all of the non-real-time ATM layer services.

The remaining ATM layer service is rt-VBR. AAL1 is not appropriate for rt-VBR. For reasons stated above, AAL5 is the simplest protocol for transmitting video. AAL3/4 provides better support for streaming data with low delay. However, AAL3/4 integrates poorly with most processor architectures [32], is more complicated than AAL5, and demands more processing and increased overhead. For this reason, AAL3/4 seems relegated to specialized applications and has been replaced by AAL5. AAL2 appears the most appropriate choice, but delays in developing the specification have slowed its employment. Choosing the correct protocol depends on the specific application and a reasonable expectation of vendor support. A more complete description of each protocol

is available in [18] except for AAL2, which is covered by ITU-T Recommendation I.363.2 [33].

6. ATM Multicast

Based on end-to-end connectivity, multimedia applications fall into three categories: point-to-point, point-to-multipoint, and multipoint-to-multipoint. Multimedia applications such as videophone or Internet telephony fall into the point-to-point category. Video on demand or remote broadcasting falls into the point-to-multipoint category. Finally, video conferencing falls into the multipoint-to-multipoint. The latter categories present a great challenge due to the need to efficiently switch video streams to avoid network loading and the additional delay added by cell duplication or readdressing [34]. The approach taken in ATM is somewhat different from multicast IP due to ATM's virtual circuit structure.

The ATM UNI 3.1 standard [23] specifies both point-to-point connections and point-to-multipoint connections. The motivation behind a point-to-multipoint connection is to conserve bandwidth by minimizing the number of VCIs required within the NNI. For example, if an end-user wishes to transmit to *N* other users, separate point-to-point connections would require *N* separate VCIs, each with the same bandwidth requirements. A point-to-multipoint connection allows VCIs to be consolidated within the NNI when they have common end-points. A point-to-multipoint VCC has the following properties. First, the multicast group resembles a tree with the sender as the root node and the receivers as leaf nodes. Second, the connection between the root and the leaves is defined by a single VPI/VCI at the UNI. Cells transmitted by the root are received by all of the leaves, assuming no losses in transmission. No bandwidth is allocated for transmission from the leaves to the root; the connection is one-way. A one-way connection is required since the root node has no mechanism for filtering data from each leaf over a single VCI[3]. Under UNI 3.1, a point-to-multipoint connection is set up as a point-to-point connection between the sender and the first leaf node. The root node then

[3] This is possible using AAL3/4 but does not appear to be practical.

adds additional leaves until the multicast group is complete. Leaf nodes may be dropped, either by their own request or by the root node, but leaves may not add themselves to the circuit. A point-to-multipoint multicast scenario is shown in Figure II.7.

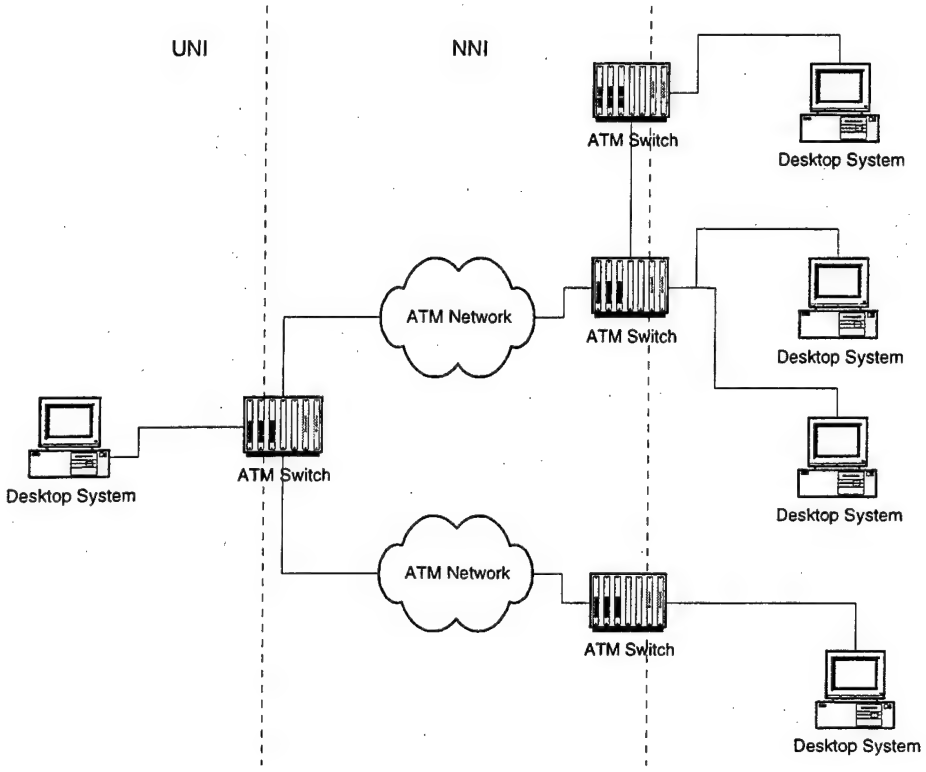

Figure II.7: ATM Point-to-multipoint Multicast.

UNI 3.1 does not provide a specification for a multipoint-to-multipoint connection. A multipoint-to-multipoint VCC has properties similar to the point-to-multipoint with an important difference. The connection is defined by a single VPI/VCI at the UNI. All cells transmitted by one endpoint of the connection are delivered to all other endpoints *and* the endpoint is capable of receiving cells over the same VCC from any of the other connected endpoints. This duplex transmission leads to several difficulties [35]. First, data cells from different sources arrive at the endpoint interleaved and must be properly reassembled by the AAL. AAL1 and AAL5 do not provide this capability [31]. AAL3/4 has a multiplexing identifier (MID) field that allows multiplexing within a VCI, but there is no standard for assigning MID values. The small

size of the MID field restricts multicast group size, and AAL3/4 requires a great deal of overhead [18][32]. The second problem is resource management. A VCC is granted only if sufficient network resources exist over the transmission path. With a multipoint-to-multipoint connection, the VCC is shared by a number of sources and determining the bandwidth requirements is difficult.

Various proposals have been made to implement multipoint-to-multipoint connections within ATM. The simplest method for implementing a multipoint-to-multipoint connection is a "forest of trees," that is, using a point-to-multipoint connection per endpoint [32][36]. With N endpoints, every endpoint is the root of a point-to-multipoint connection with $N - 1$ leaves. A "forest of trees" offers low latency per network node, but a member entering or exiting from the multicast group causes a burst of signal messages. This approach is specified by the ITU-T H.3XX multimedia conferencing standards. Another approach is to use a server as an intermediary [37]. Each endpoint transmits data over a point-to-point connection with the server. The server relays the data to the other endpoints through a point-to-multipoint connection for which it is the root node. The Shared Many-to-many ATM ReservaTions Protocol (SMART) [35] is a novel ATM layer level protocol that regulates access to the multicast tree. SMART requires only one VCC for the entire multicast group although more VCCs are allowed to support concurrent data transfer by two or more endpoints. Access to the shared VCC is provided by a grant mechanism implemented in a round-robin fashion. The SMART protocol has proven viable for multicast VTC traffic with suitable modifications to the grant mechanism to account for the needs of real-time traffic [38].

C. WIRELESS NETWORKS

The typical military wireless network is based on packet-radio technology that extends the concept of the point-to-point packet-switched network to a broadcast radio medium. Like some LAN standards, such as Ethernet, the radio channel is inherently a multiple-access medium that provides a much less reliable transfer medium than that experienced in wireline networks. As shown in Figure II.8, the data link control (DLC)

layer provides service to higher layer protocols, such as IP and ATM, by transferring data in packets or cells over the radio medium. The DLC specifically provides reliable transfer of information across the physical link and regulates access to the shared medium. The functionality of the DLC is separated into the logical link control (LLC) and medium access control (MAC) sublayers. While the functionality of the layers shown in Figure II.8 is described briefly below, a more thorough discussion of packet-based radio networks can be found in [39].

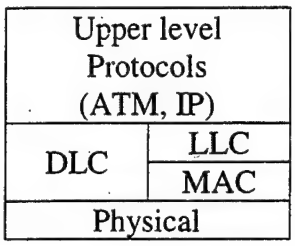

Figure II.8: DLC for a Packet-Based Radio Network.

1. Logical Link Control

The LLC layer provides an interface to the network layer, either IP or ATM for example, and performs error and flow control. Error control involves providing mechanisms for responding to errors in transmitted frames while flow control regulates the flow of frames to ensure the sender does not overwhelm the receiver. Errors occur due to bit or burst errors during transit, which either damage the frame or cause the frame to be unrecognizable. Error control is usually provided by an automatic repeat request (ARQ) mechanism that combines error detection from the MAC with positive and negative acknowledgements and retransmission after timeout. For real-time traffic, the viability of the ARQ mechanism depends on the overall delay budget, and the LLC may confine itself to dropping the corrupt data packets. The LLC layer may also attempt to correct errors if forward error correction (FEC) coding is employed. Another possibility is to perform power management at the LLC layer to vary transmission power in response to observed error rates.

2. Medium Access Control

The MAC governs access to the transmission medium, performs conflict resolution and provides error detection. A MAC protocol is either centralized, where a controller grants access to the network, or decentralized wherein all stations dynamically determine access. Various protocols are available to control access including round robin or polling, reservation, and contention. With polling protocols, each station is given an opportunity to transmit in turn. Reservation schemes are more suitable for stream traffic and divide access time into slots, which allows stations to reserve slots when data is ready for transmission. Contention schemes work well for bursty traffic where all stations attempt to seize control of the medium and backoff when collisions occur. Contention works well only for light-loaded networks. Of the three schemes, reservation provides the greatest throughput and least delay for integrated wireless networks. Slot-based reservation schemes for wireless ATM networks and mobile IP networks have been proposed by [39] and [40], respectively.

Referring back to Figure II.8, information flows in the flowing manner. The network layer passes cells or packets to the LLC. The LLC appends a control header, creating an LLC-PDU. The control header provides the data required for flow control and error control. The LLC-PDU is passed to the MAC, which assembles a frame containing one or more LLC-PDUs along with address and error detection fields. Once access is granted to the radio medium, the frame is transmitted in order by the physical layer.

3. Physical Layer

The physical layer specifies the transmission medium, signal encoding, synchronization, and bit transmission/reception. Although the MAC layer determines access to the channel, a wideband radio channel may be segregated several ways [41]. The simplest is time division multiple access (TDMA) in which a sender transmits during a fixed time slot. The channel may also be split into several independent, smaller channels using frequency division multiple access (FDMA) or code-division multiple

access (CDMA) to allow multiple users to transmit simultaneously. Finally, TDMA may be combined with either FDMA or CDMA.

D. LAYERED VTC OVER ATM

1. ITU-T Multimedia Standards

The ITU-T H-series recommends several standards for real-time multimedia communications, each targeting a different network architecture. The standards proposed for ATM networks are briefly reviewed to provide some motivation for the layered VTC over ATM implementations proposed in this dissertation.

Each ITU-T H-series multimedia conferencing standard associates a set of video, audio, multiplex, and control standards into a multimedia terminal [42]. Each terminal provides point-to-point, real-time audio and video conferencing at various levels of quality with provisions for optional data transfer. Data transfer possibilities include graphics, still images, and control signals such as those needed for remote camera operation. Extensions to the base standards allow multipoint operation and encryption with appropriate network support. The ITU-T standards have found wide acceptance, and hardware implementations are readily available in PCI and compact PCI card formats. Two ITU-T standards address ATM networks: H.321 and H.310.

H.321 is a first generation standard and adapts the earlier H.320 recommendation to ISDN networks [42][43]. As expected from a standard adapted from ISDN networking, H.321 allocates bandwidth in increments of 64 kbps. The baseline video codec specified is H.261, which compresses color video at a constant bit rate in increments of 64 kbps. H.261 supports two resolutions: CIF (352×288 pixels) and QCIF (176×144 pixels). Baseline audio is compressed using the G.711 log-PCM codec, providing low-delay, toll-quality narrowband audio at 64 kbps. H.321 uses the AAL1 protocol to support data channels equivalent to ISDN 'B' channels by mapping one 'B' channel per VCC.

H.310 is a native standard for videoconferencing over ATM/B-ISDN and includes the earlier H.321 as a subset [42][44]. Figure II.9 gives a simplified functional

description of a H.310 terminal and associated standards for multiplexing, call establishment, and data transfer. Taking advantage of the high bandwidth available in B-ISDN networks, H.310 offers high-quality video using the MPEG-2 video codec and high-quality audio using Layer II MPEG-1 audio. To support H.321 terminals, H.261 video and G.711 audio are also supported with H.263 video, a codec optimized for low bit rate channels such as analog modems, as an option. H.310 terminals support a variety of data rates, but all terminals are required to support common rates of 6.144 and 9.216 Mbps. Calls are established by creating an initial VCC to set up a control channel. This control VCC uses the AAL5 protocol. Once two terminals have established a set of operating parameters, a second VCC is created to carry multiplexed audio and video. Either the AAL1 or AAL5 protocol is used. Additional VCCs may be established to carry data traffic.

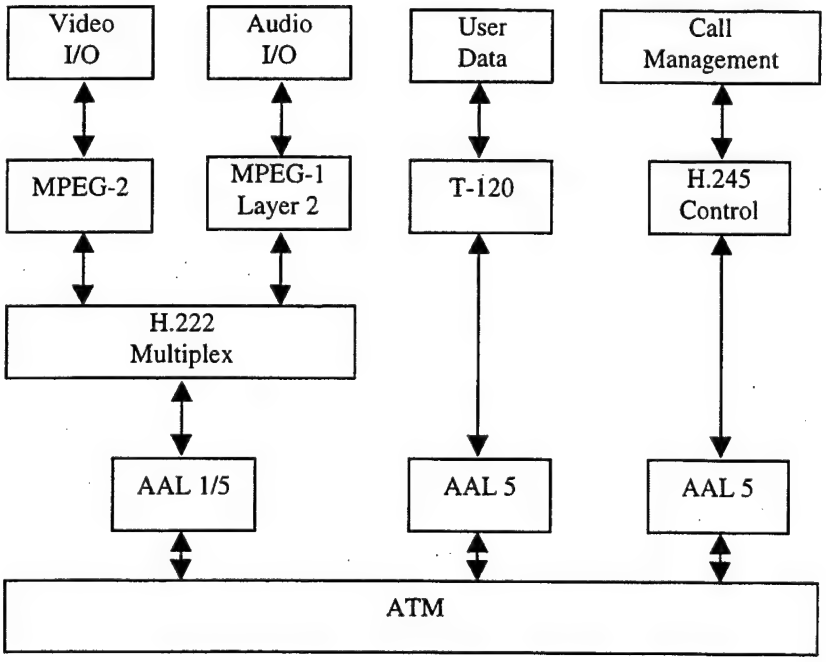

Figure II.9: ITU-T H.310 B-ISDN Terminal.

2. Layered Video Considerations

Compared to the ITU-T terminal recommendations, layered video poses a different set of considerations in determining a feasible network interface. Chief among

these is the desire to enable the ATM layer to discern which video layer owns an individual cell. Associating layers with individual cells allows an ATM switch to exploit the hierarchical nature of layered video through scheduling to actively control congestion while maintaining the best possible end-to-end video quality. Another benefit is offering recipients the ability to subscribe to any number of layers they initially choose as well as a means to add or drop layers during the session. This is the core promise of RLM [45].

A secondary concern is to allow the network to identify logical elements within the video stream, such as the frame header and group-of-block (GOB) boundaries (see Figure III.1). Locating GOB boundaries provides another dimension to network scheduling by allowing the switch to identify cells that will not aid video reconstruction at the recipient due to previous cell losses (see Chapter VI). Two approaches for allowing identification of video layers at the ATM layer are proposed here. The first is to assign each video layer to a separate VCC. The second requires multiplexing individual layers over a single VCC. Each approach impacts the network interface design differently: the most appropriate AAL protocol, schemes for manipulating the ATM cell header, and the manner in which the multipoint-to-multipoint connection is established. GOB identification is considered only briefly here; more details are provided in Chapter VI. No attempt is made to provide a complete multimedia terminal specification such as H.310. Instead, the goal is to demonstrate the feasibility of supporting layered video within existing ATM standards.

In addition to the layering scheme, the choice of AAL protocol depends on the services required by the application. Here, we assume that the audio and video streams are not multiplexed as they are in H.310. Segregating the streams allows different service for audio and video and simplifies network scheduling with respect to the layered video.

We first consider the audio stream. The tactical scenario requirements (see Table I.1) limit the audio stream bit rate to 8 kbps. The G.711 and MPEG-1 Layer 2 codecs are obviously incompatible with the scenario requirements. This is not surprising since H.310 targets B-ISDN. However, other high-quality narrowband audio codecs are available that specifically target low bit rates. Two suitable codecs specified in the H.324

recommendation for low-bit-rate circuit-switched networks, such as the PSTN, are the G.723.1 and G.729 codecs. G.723.1 transmits at either 5.3 or 6.4 kbps and offers near-toll-quality speech although codec delay is rather large for VTC applications [5]. G.729 offers higher quality and lower coding delay for a similar level of complexity. Both codecs offer silence detection to reduce bit rate by either not transmitting or transmitting only background noise. Of the two, G.729 appears the best choice for the tactical scenario considered here. Given that G.729 transmits at a fixed-bit-rate, the AAL1 protocol appears to be best suited.

The question for the video stream is not which codec to use, since a layered coder is assumed, but the type of rate control to employ. Three options are possible: CBR, VBR with no constraints, and VBR with bit-rate constrained to a predetermined average. Assuming a fixed quantization scheme at the encoder, compressed video is naturally VBR since compression gain varies frame-to-frame. Bit rate constraints come at the cost of quality variations [46]. CBR tends to show larger fluctuations in visual quality relative to VBR and may be unappealing at low bit rates. VBR with a predetermined mean-bit-rate demonstrates quality fluctuations between VBR and CBR. As indicated above, VBR streams have another advantage in that bandwidth can be conserved through statistical multiplexing, a significant advantage in low bit rate networks. However, resource allocation is simpler if the mean bit rate is constrained since ATM traffic descriptors, such as PCR and SCR, are easier to determine. For these reasons, the video stream is assumed to be VBR constrained to a predetermined mean bit rate. Only the AAL1 protocol is rendered unsuitable by this assumption and choosing among the remaining protocols depends on limitations introduced by video layering as discussed below.

The last issue to consider is that of synchronization of the audio and video streams. We assume that if the application is given suitable timing information for each stream, then it is capable of synchronizing playback. Timing information is either provided to the application by the AAL or determined directly using time-stamps embedded in the application PDU. The former approach is available only if AAL1 or AAL2 is used. The latter is offered by encapsulating application data within a RTP

packet. Each RTP packet includes a 32-bit timestamp corresponding to the time when the first octet of data was generated. The exact approach taken in this work is outlined below.

3. Multiple VCC Case

In the multiple VCC approach, each video layer is assigned a separate VCC and is readily identified within the network by its VPI/VCI pair. For scheduling purposes, a switch needs to logically associate the VPI/VCI pairs transporting the video layers from a particular sender and to establish a hierarchy for priority service. A simple means of logically associating layers is to assign one VPI per sender or to negotiate VPI/VCI pairs in contiguous blocks[4]. Using multiple VCCs conveys several advantages. Using individual VCCs allows a great deal of flexibility in providing service on a per-layer basis. The sender can negotiate different service and different QoS for each individual layer, even in the absence of a dedicated scheduling algorithm for layered video. Multiple VCCs also simplifies the task of allowing end users to subscribe to individual layers at call setup and dynamically add or drop layers once the VTC is in progress. A penalty is paid due to the large number of connections. Call setup time is increased and changes to the multipoint-to-multipoint connection incur a proportionate increase in signaling amongst the end-points.

Service is provided to each layer using the AAL5 protocol. AAL5 offers the lowest overhead of the VBR protocols, eight octets per CS-PDU and no additional overhead in the SAR-PDUs. It is also the most appropriate choice if a higher-level protocol, such as RTP, is employed.

Data transfer proceeds as shown in Figure II.10. The video compressor relays application PDUs over to the AAL after time-stamping each to facilitate synchronization with the audio layer. An application PDU consists of a single GOB, multiple GOBs or an entire frame. The choice depends on the manner in which frame elements are exploited by the coder. In the CS sublayer, an eight-octet trailer is appended, and the CS-PDU is

[4] Negotiating VPIs and/or VCIs is not supported in UNI 3.1 but is supported by UNI 4.0 [47].

padded out to a multiple of 48 octets. The trailer includes the CPCS user-to-user indication field, which allows transparent transfer of user information between end-users or application layers. The user-to-user indication field identifies the video layer (0 = base, 1 = first enhancement layer, and so on), which enables the end application to associate each incoming VCC with a layer and correctly reassemble the video stream. The SAR sublayer segments the CP-SDU into 48-octet SAR-PDUs; no headers or trailers are necessary. At the ATM layer, each SAR-PDU is encapsulated into an ATM cell information field.

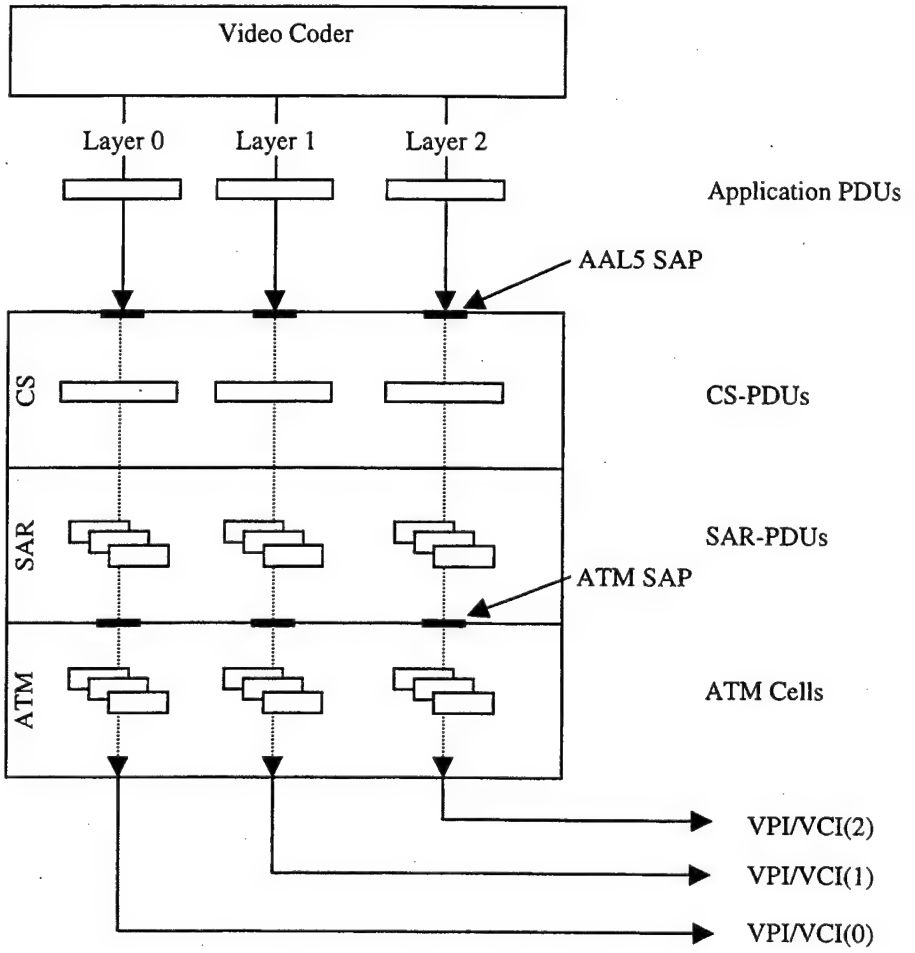

Figure II.10: Transmitting Layered Video Using AAL5 and Multiple VCCs.

Since the AAL5 SAR merely segments the CS-PDU, the endpoint CS sublayer cannot distinguish between SAR-PDUs containing the CS-PDU payload and the SAR-

PDU containing the trailer that ends the CS-PDU. To distinguish between these cases, the SDU-type bit in the payload type field is used. At the ATM layer, a CS-PDU consists of zero or more ATM cells with the SDU-type bits set to zero followed by an ATM cell with the SDU-type bit set to one. The latter indicates the presence of the CS-PDU trailer and the end of the CS-PDU. This scheme also allows the network to determine the boundaries of the application PDU by tracking changes in the SDU-type bit. Figure II.11 shows how a GOB, assuming that the application PDU consists of a single GOB, is located within the ATM cell flow. Therefore, a scheduling algorithm could track the SDU-type bit to incorporate GOB boundaries into scheduling decisions.

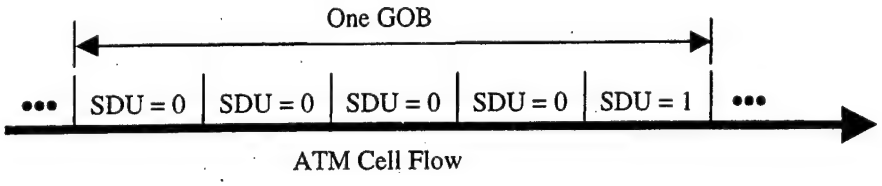

Figure II.11: Use of the SDU Bit to Locate Application PDU Boundaries with AAL5.

Establishing a multipoint-to-multipoint connection follows the procedures outlined under ATM multicast above with the difference that a separate point-to-multipoint connection must be established for each layer. The order in which connections are established is potentially of importance if the network possesses limited resources over any path that forms part of a connection. To preserve the hierarchical nature of video layering, the first point-to-multipoint connection established should be the VCC associated with the base layer. In turn, VCCs associated with the enhancement layers are established, one by one, in order of each layer's perceptual importance. While establishing a complete set of connections in this manner entails a longer setup time than negotiating each connection simultaneously, a hierarchical connection order prevents lack of resources from denying a connection to a more perceptually important layer in favor of a less important layer. Therefore, the network arbitrates which layers receive connections based on the resources present over all paths composing the point-to-multipoint connection. If an endpoint workstation does not possess the capability to decode all the layers comprising the video session, the workstation can refuse connection to unwanted

layers. The individual endpoint should also deny connection in the case of an illegal layering arrangement. This may happen if the network does not permit a connection for a layer while a less important layer is allowed to establish a connection due to smaller bandwidth demands.

4. Single VCC Case

The case for limiting the layered video stream to a single VCC is driven by the desire to minimize the number of active connections in the multipoint-to-multipoint connection. While VCIs are not a scarce commodity – a single VPI can bundle as many as 65536 VCIs with the values 0-32 reserved [23] – signaling and control requirements increase with the number of connections, which subsequently increases call setup time. An alternative approach is to multiplex cell flows from each layer within a single VCI. Multiplexing flows over a single VCI is only supported by AAL2 and AAL3/4. Since AAL3/4 has been largely replaced by AAL5, the problem of supporting a single VCC rests on determining a suitable interface between the application layer, the AAL2 protocol, and the ATM layer.

Unlike the other AAL protocols, AAL2 specifies only a CS sublayer and does not utilize a SAR sublayer [33]. The CS sublayer functionality is further split into service-specific (SSCS) and common parts (CPCS) sublayers. The simplest SSCS definition is the null SSCS which transfers application PDUs directly to the CPCS sublayer. Other definitions remain under study, and a SSCS definition for layered video traffic is proposed below. The CPCS sublayer multiplexes individual cell flows and provides VBR traffic support.

The following service approach is proposed to adapt AAL2 for layered video. Referring to Figure II.12, each layer is assigned a service access point (SAP) at the AAL SSCS sublayer. The application PDU consists of a GOB, a contiguous set of GOBs, or a frame from a particular layer. The application PDU is buffered within the SSCS sublayer and transmitted in blocks to the CPCS sublayer. Block size is set at 44 octets to increase transmission efficiency. If the application PDU length is not a multiple of 44 octets, a variable length block is transmitted with length < 44 octets.

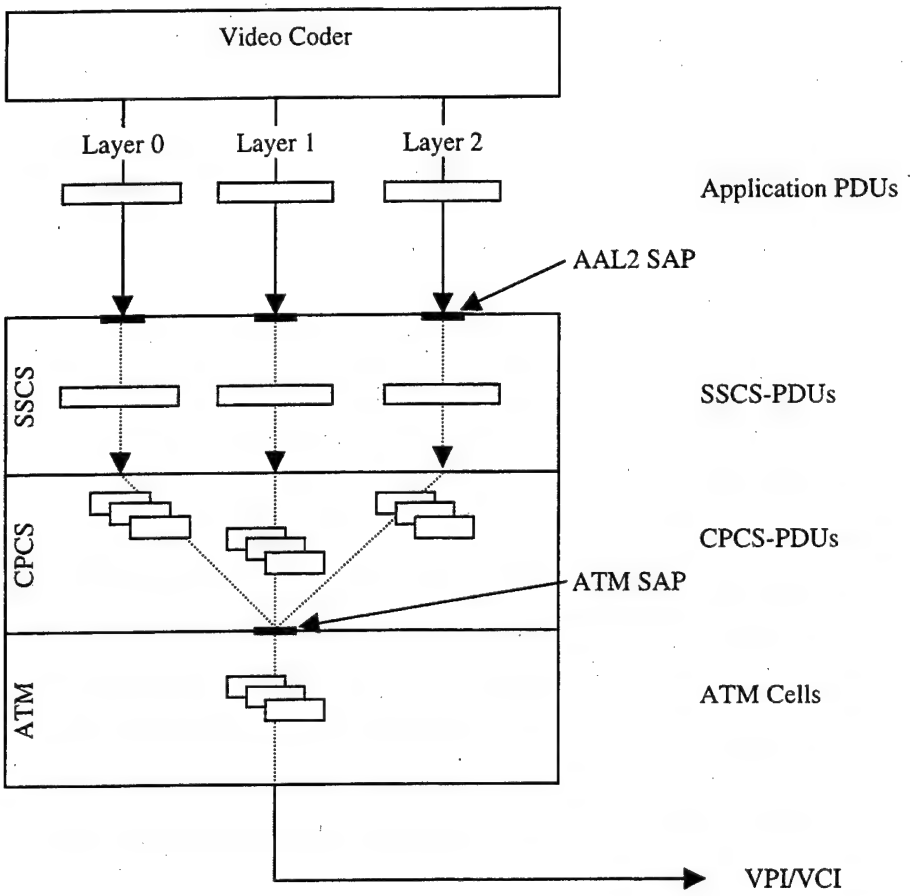

Figure II.12: Transmitting Layered Video Using AAL2 and a Single VCC.

The CPCS sublayer accepts blocks from each SSCS SAP and appends a three octet header to form a CPCS packet. Within the header, the Channel Identifier (CID) uniquely identifies the layer number. The CID field is 8 bits in length which, after allowing for reserved values, permits identification of up to 248 individual channels. Since available channel numbers start at 8, one possible scheme is to start numbering channels with $CID = 8 + layer\ number$, where layer numbers start at zero for the base layer. The length indicator field is set to reflect either a fixed payload length of 44 octets or a smaller, variable value for the last segment in the application PDU if the application PDU is not an even multiple of 44 octets. The CPCS packet is then loaded into a CPCS-PDU with an 8-bit start field header. If the length of the last CPCS packet is less than 47

octets, a trailer is added to pad the CPCS-PDU to 48 octets. The combined overhead of the CPCS packet header and the CPCS-PDU start field header is exactly four octets. Therefore, a block size of 44 octets at the SSCS sublayer simplifies processing by the AAL since each CPCS packet and associated CPCS-SDU is transported within exactly one ATM cell.

An alternate approach that reduces overhead is to buffer application-PDUs at the SSCS sublayer. Each application-PDU is segmented into 44-octet blocks as before and transmitted to the CPCS sublayer. If an application-PDU is not an even multiple of 44 octets, the leftover bits are retained at the head of the SSCS buffer. When the next application PDU is buffered and segmented at the SSCS sublayer, data from the last application-PDU is encapsulated into the first CPCS packet. Although this approach transmits data from different application-PDUs in the same ATM cell, overhead is reduced considerably since every CPCS packet is filled to 44-octets, obviating the need to ever pad the CPCS-SDU.

At the destination AAL, the CPCS sublayer strips the SF header off the CPCS-PDU and reads the CID field within the CPCS packet header to route the payload appropriately to the SSCS sublayer. No specific functionality is envisioned for the receiver side of the SSCS sublayer. The SSCS sublayer merely accepts the payload from the CPCS sublayer and forwards it to the application layer. There is no need to recreate the application PDU since the decoder is assumed to be capable of interpreting the raw bit stream.

The above approach allows the cell flows of each layer to be multiplexed over a single VCC. However, the network is unable to distinguish between the different flows if the only indication lies within the ATM cell information field. As ATM switches only read cell headers, layer designation must occur using fields within the cell header as shown in Figure II.4. By design, ATM cell headers are relatively small, incorporating only the information required for ATM switches to perform their switching and congestion control functions. Therefore, the sender has very little flexibility in setting individual fields within the header that are not subject to being overwritten by switches.

However, the SDU-type bit and the CLP bit are available to the user [23]. Used together, the two bits allow indication of up to four layers (although only three layers are employed here) as indicated in Table II.3. The CLP bits are enabled for the lower priority layers. Setting the CLP bit does not necessarily indicate cells from enhancement layers are automatically dropped during periods of congestion. The user is allowed to negotiate QoS separately for the cell flow consisting of cells with the CLP bit set to zero and the cell flow consisting of all cells (CLP = 0/1) [28]. Setting the CLP and SDU-type bits requires extending AAL2 to communicate with the ATM layer in a manner similar to the interaction between AAL5 and the ATM layer. A method to accomplish this is to transfer the CID field value with the CPCS-PDU. The ATM layer uses the CID value to determine an index into Table II.3, $index = (CID - 8)$, and sets the CLP and SDU bits appropriately.

Layer Number	SDU bit	CLP bit
0	0	0
1	1	0
2	0	1
3 (not used)	1	1

Table II.3: ATM Cell-Tagging Scheme for Layered Video.

In the multiple VCC case, the SDU-type bit is available and enables the network to determine the application PDU boundaries in order to incorporate logical video elements such as a GOB or frame into scheduling decisions. The cell-tagging scheme presented in Table II.3 does not permit a similar approach at the network level. An alternative approach requires the AAL to segregate CPCS-PDUs resulting from each layer's application PDUs. The segregated CPSC-PDUs are then handed to the ATM layer and transmitted sequentially. Since each CPCS-PDU comes from the same channel, an application PDU appears to the network as a contiguous set of cells, each with the same cell-tags. By monitoring changes in the CLP and SDU-type bits, the network can identify application PDU boundaries. This approach is shown in Figure II.13. While

convenient, concatenating application PDUs within the VCC impacts scheduling performance. This issue is covered in more detail in Chapter V.

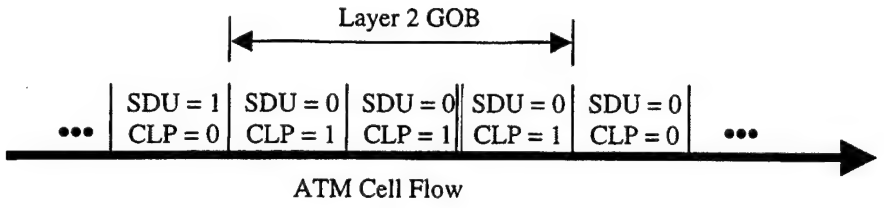

Figure II.13: Identifying Application PDUs in a Multiplexed Cell Flow.

Setting up a multipoint-to-multipoint connection requires each sender to establish separate point-to-multipoint connections for the audio and video streams. Compared to the multiple VCC approach, creating and maintaining a VTC session with a single VCC reduces signaling requirements. However, using a single VCC reduces flexibility in heterogeneous networks. When the initial connection is established, the sender must negotiate acceptable QoS for the entire video stream. While this appears to negate the flexibility offered by transmitting layers, the sender still has the option of negotiating QoS separately for the CLP = 0 and CLP = 0 + 1 cell flows. For similar reasons, individual endpoints cannot refuse individual layers at call setup and must accept the entire video stream or decline the connection. Still, it is desirable to allow an endpoint to dynamically drop layers, both to ensure that the more important layers arrive and to reduce bandwidth demands within the network if no downstream nodes require certain layers. Chapter VI proposes a scheme that allows the network scheduler to effectively drop individual layers within a VCC when no destination indicates an interest in those layers.

This chapter examined architectures suitable for transporting real-time, interactive multimedia information streams. A suitable network architecture needs to meet the following requirements: multicast support, QoS guarantees, and real-time support. The ensuing discussion indicated that only ATM networks currently meet all three requirements. Given that ATM is a viable networking architecture, two approaches are

presented to transmit layered video. The first approach assigns each layer to a separate VCI using AAL5. This approach is the most versatile in allowing network access to individual layers; it scales well and provides easy access to GOBs within each layer. The primary drawback is the increased signaling in a multicast scenario since each individual connection represents the base of a multicast tree. The second approach multiplexes each layer across a single VCI using AAL2. This approach offers quicker call setup and minimizes signaling in multicast scenarios but requires modification to the CPCS sublayer to tag each cell with an appropriate identifier for each layer. On the other hand, a single VCI cannot scale beyond four layers, and organizing the stream into recognizable GOBs is somewhat complicated.

III. VIDEO CODING TECHNIQUES

Even when considering the modest requirements outlined for the video teleconferencing scenario presented in Chapter I, raw video signals are very bandwidth intensive. Consider an example using the specifications listed Table I.1 with gray-scale video only. Sending an uncompressed grayscale video stream at 8 bits per pixel requires a bandwidth of approximately 2 Mbps; this is not an insurmountable requirement with a dedicated wireline ATM network but clearly excessive for tactical video teleconferencing. Restricting the video stream to an average of 64 kbps requires a compression gain of about 31 to 1 or an average bit allocation of 0.26 bits per pixel (bpp). Transmitting a true-color vide sequence over the same channel would require a compression gain three times higher.

This chapter presents a basic discussion of hybrid video coding and includes transform coding, motion compensation, quantization, and entropy encoding. A quick measure for quantifying distortion due to quantization is introduced as a measure of picture quality. The MPEG and H.263 video coding standards are described and examined for error resilience. Finally, wavelet-based image compression is presented in preparation for the layered video discussion in the next chapter.

A. VIDEO COMPRESSION OVERVIEW

Video coding involves a combination of removing perceptually redundant content, representing information efficiently through lossless coding, and exploiting frame-to-frame correlation within a video sequence. Motion video is typically low-pass in nature; the human eye places greater relative weight on lower frequencies than higher frequencies [6]. Therefore, 2-D transform methods are used to generate an equivalent frequency domain representation, a process that is lossless and invertable. Using this representation, variances in human perception are exploited by quantizing the resulting coefficients to different degrees of precision with more precision granted to the lower

frequencies. Quantization reduces the dynamic range of the coefficients, which results in information loss but enables the coefficients to be represented with fewer bits. Usually, the least relevant coefficients are zeroed out during quantization, thus creating runs of zeros. Since there is little need to explicitly represent the zeros, run-length coding is used to generate a more compact representation that is, in turn, replaced by a more efficient, lossless variable-length coding (VLC). Taken collectively, these techniques are referred to as spatial compression and form the basis of image compression standards, such as JPEG.

A video codec must compress a time-varying video sequence consisting of a series of frames spaced at equal time intervals. The codec may or may not exploit the temporal dimension depending on the application requirements. The simplest approach is to ignore any correlation between individual frames and compress each frame independently as if it were a still image. This approach is known as intraframe coding, and the resulting compressed frames are referred to as I-frames. An example is Motion-JPEG, which uses JPEG to code individual frames. Intraframe coding offers the advantage of error resilience since decode errors are confined always to the current frame. However, compression gain is limited to about 0.5 bits/pixel with acceptable image quality [6]. Higher compression gains are possible, for the same quality, by exploiting the high degree of correlation that video frames tend to exhibit from frame-to-frame. Interframe coding removes redundancy by only coding the differences between successive frames. When these differences arise due to motion, interframe coding yields compression gains that vary in relation to the degree and type of motion. Static frames exhibit a high degree of compression while rapid motion tends to degrade compression performance. The drawback to interframe coding is the dependence between successive frames at the decoder. If errors occur in the current frame, the errors tend to propagate temporally between successive frames as well as spatially within the frames. Of course, if two successive frames are not correlated, perhaps due to a scene change, interframe coding performs no better – typically worse due to additional overhead – than intraframe

coding [7]. Therefore, video codecs, such as H.263 and MPEG, incorporate both types of coding for efficiency and, in some cases, to place an upper bound on error propagation.

B. VIDEO CODING HIERARCHY

To facilitate different aspects of video coding and decoding, the video stream is organized into a hierarchy of logical elements. The organizational scheme varies from coder to coder, but the most common elements are presented below.

The basic display unit is the picture or frame and is comprised of rectangular array of pixels, which in turn represent data structures indicating the color and luminosity of each pixel. The dimensions of the array represent the picture resolution, given as columns × rows, where the codec of choice determines the available resolutions. A set number of contiguous pictures are organized into a group of pictures (GOP). A GOP usually influences compression gain and consists of an intraframe coded picture followed by a series of interframe coded pictures.

Within a frame, pixels are organized, in order of increasing size, into blocks, macroblocks, and groups of macroblocks (GOB) or slices. A block is an 8×8 array of pixels and is the basic element for transform coding operations, such as the discrete cosine transform (DCT). Motion compensation is applied at the macroblock (MB) level, a 16×16 array of four blocks, to reduce the associated overhead and computational expense. A frame may be viewed as being composed of rows of macroblocks. For example, a frame with a resolution of 176×144 pixels contains nine rows of macroblocks with eleven macroblocks per row. One or more contiguous rows of macroblocks are termed a GOB or a slice depending on the codec. GOB is the more general term while the term slice is defined within the MPEG-1/2 standards [6]. GOB headers, along with the frame header, serve as reference points that allow the decoder to resynchronize with the incoming bit stream after decode errors caused by lost packets or bit errors. A representation of the hierarchy superimposed on the compressed bit stream is shown in Figure III.1; the length of each compressed frame varies due to variable compression gains.

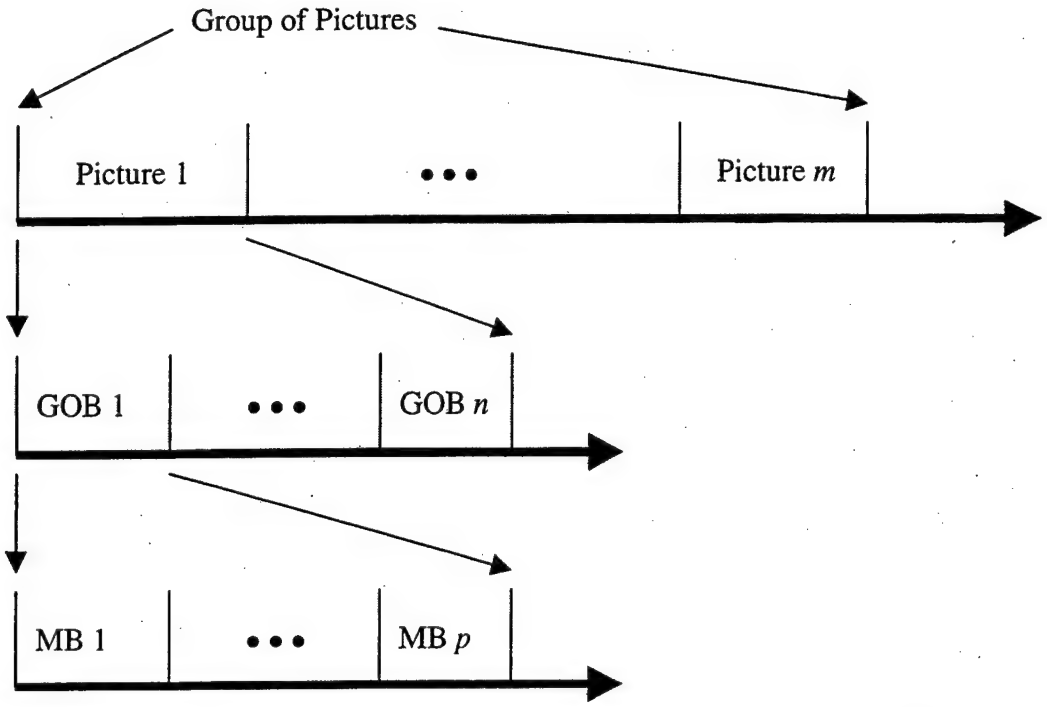

Figure III.1: Organizational Hierarchy for Compressed Video.

C. INTRAFRAME CODING

Intraframe coding (or spatial compression) is essentially the same as still image compression. Each frame is compressed independently by removing redundant information within that frame, balancing compression against image quality, and coding the remaining information in a more efficient manner. No attempt is made to exploit temporal correlation existing between frames. The three steps comprising intraframe coding are shown in Figure III.2 and are explained further below.

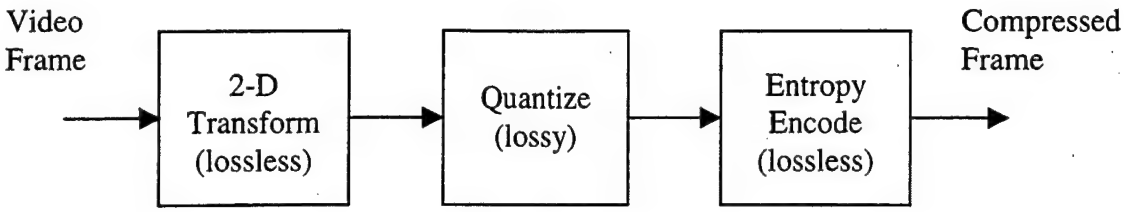

Figure III.2: Overview of the Steps Comprising Intraframe Coding.

1. Transform Coding

A frame represents a sampled version of the original scene at a single instant in time. Contiguous regions of samples (or pixels) tend to be highly correlated, and in practice compression through direct scalar quantization is inefficient[5]. Instead, application of a suitable linear transform to decorrelate the samples gives a greater level of compression for a given encoder complexity [48].

A suitable transform increases compression efficiency as follows. A signal is decorrelated if application of the transform results in diagonalizing the signal's autocorrelation matrix. Equivalently, the resulting transform coefficients are not correlated. An optimal transform tightly packs energy into the smallest number of coefficients possible, a property known as "energy packing" efficiency [48]. The advantage is that if the coefficients are arranged in decreasing order of magnitude, retaining only the first k out of N coefficients gives the least distortion as measured by MSE. The advantage is that, although the transform is lossless, a given level of quantization results in the least distortion of the original data.

Another advantage of transforms is that the new domain is often more appropriate for perceptual-based quantization. Certain transform coefficients may hold greater perceptual relevance. For example, the human visual system (HVS) places the most importance on low frequency details in images or video [6]. This dependency may be

[5] Still, direct techniques are employed where lossless compression is the primary concern.

exploited using frequency-based transforms and then distributing quantization errors in relation to the relative importance of each coefficient.

In theory, the discrete-time Karhunen-Loeve transform (KLT) provides the greatest energy packing efficiency [49]. However, the KLT is both computationally intensive (order of N^2) and signal dependent, thus requiring a separate eigenvector calculation for each transformed data block. These liabilities preclude the use of the KLT in video compression. Instead, video coders use transforms that approximate the KLT's energy packing efficiency and possess more efficient algorithms.

The most widely used transform for image processing is the two-dimensional discrete cosine transform (DCT). The DCT provides the closest energy packing performance to the KLT, and numerous fast algorithms are available, frequently implemented in hardware, that reduce the computational effort to the order of $N\log_2 N$ [6]. For example, a 2-D DCT can be implemented with as little as 54 multiplication operations [50].

A frame is transformed by dividing its elements into $N\times N$ blocks of pixels and applying the 2-D DCT to each individual block. The typical block size is 8×8. Larger block sizes are possible, but the pixels tend to be less correlated, which decreases the resulting compression gain. Denoting the original block as $f(i,j)$ and the transformed coefficient block as $F(u,v)$, the 2-D DCT is given by [6]

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) \cos\left(\frac{(2i+1)\pi u}{2N}\right) \cos\left(\frac{(2j+1)\pi v}{2N}\right) \qquad \text{(III-1)}$$

where u and v are the horizontal and vertical frequencies, respectively, and

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0, \\ 1 & \text{otherwise.} \end{cases} \qquad \text{(III-2)}$$

The inverse DCT is given by

$$f(i,j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\left(\frac{(2i+1)\pi u}{2N}\right) \cos\left(\frac{(2i+1)\pi v}{2N}\right). \qquad \text{(III-3)}$$

Transforming an 8×8 block of pixels results in a block of 64 coefficients with a spatial frequency distribution as shown in Figure III.3. The $F(0,0)$ coefficient represents

the DC value while the remaining coefficients are termed AC coefficients. Figure III.4 indicates how images elements map into the frequency domain via the 2-D DCT [6]. Individual blocks within a frame tend to show little variation from pixel to pixel, an indication of low-pass frequency content. Given this condition, the magnitude of the DCT coefficients is largest in the region about the DC coefficient and diminishes with increasing frequency.

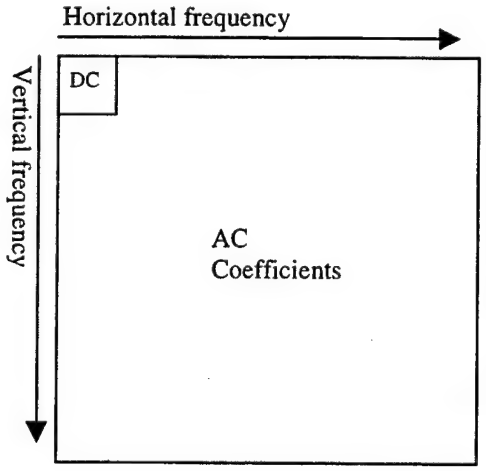

Figure III.3: Frequency Interpretation of DCT Coefficients.

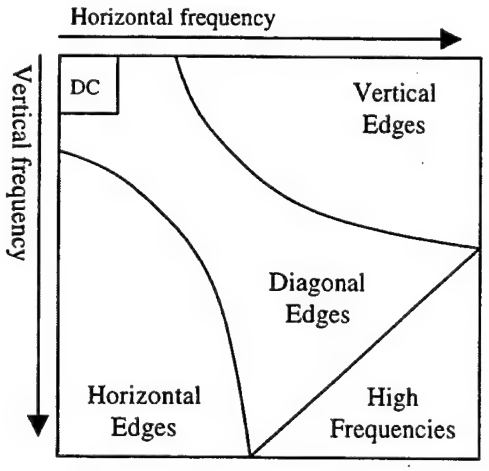

Figure III.4: Structural Decomposition of Image Elements [6].

The need for data blocking in DCT-based compression becomes a liability with high levels of compression. Compression tends to remove high-frequency components,

which leads to smoothing of the visual content of each block and creates "blocking artifacts" that disturb the continuity of the frame. The same effect also leads to the presence of "ringing" artifacts around sharp edges [3].

2. Scalar Quantization

The DCT coefficients are quantized to reduce precision, which allows each coefficient to be represented with fewer bits. Quantization may also remove the least significant coefficients by setting their value to zero. The tradeoff is added quantization noise, which shows up as distortion within the reconstructed image. The most typical quantization scheme employed is uniform quantization wherein each coefficient F_{uv} is divided by the quantizer step size Q_{uv} and the result rounded to the nearest integer as follows [5]:

$$F_{quv} = \text{round}\left(\frac{F_{uv}}{Q_{uv}}\right), \forall u, v. \qquad \text{(III-4)}$$

The reconstructed value is found by multiplying the quantized coefficient by the quantizer step value, $F_{quv} \times Q_{uv}$. As Eq. (III.4) implies, the quantizer step value may vary with each DCT coefficient as discussed below. In this case, Q_{uv} represents an element from an $N \times N$ quantizer matrix. Alternatively, a single value may be used for the entire block for simplicity. Although uniform quantization is widely used, the choice is not optimal since analysis has shown that individual coefficients are not distributed uniformly [51]. Other approaches have been suggested to reduce the quantization error, such as employing a separate Max-Lloyd quantizer for each coefficient [52], but the gain does not appear to outweigh the computational effort.

Since not all coefficients are significant, some may be discarded prior to quantization [6]. In maximum variance zonal sampling, the coefficients are ordered by the magnitude of their variance and a fraction of the N^2 coefficients with the largest variances are retained with the remaining coefficients set to zero. Threshold sampling

performs the same function but retains coefficients on the basis of the largest magnitude [6].

However, the most common approach is to weight the relative importance of each coefficient by careful selection of quantizer step values Q_{uv}. Small quantizer step values yield less distortion but require more bits. Larger quantizer step values introduce larger distortion but tend to result in more zeros and require fewer bits. Choosing the optimal step size requires selecting a suitable criterion, either through a bit-allocation approach or human visual system (HVS) modeling. In bit-allocation, the magnitude is chosen to minimize distortion within a bit budget for the block or frame. One optimal scheme varies each quantizer in proportion to the variance of the coefficient, which yields the same average distortion for each coefficient [48]. However, bit allocation schemes fail to account for human sensitivity to different spatial frequencies. Instead, most international coding standards, such as JPEG and MPEG, employ quantizer matrices based on HVS models. Using HVS models as a reference, the quantizer step sizes are chosen such that lower frequency coefficients are quantized more finely while higher frequency coefficients are quantized more coarsely [6]. The HVS is also more sensitive to luminance intensity than chrominance, so different quantizer matrices are developed for each.

A desirable feature in video encoders is the inclusion of rate control for the outgoing compressed video stream since each frame's compression gain depends on the frame's contents. For example, the encoder may attempt to maintain a constant bit rate or a constant average bit rate, or to allow bit rate to vary without constraint. Control is exercised by varying video quality to achieve the desired bit rate. Referring to Figure III.2, only the quantizer introduces distortion and affects the reconstructed quality of the frame. Therefore, rate control schemes use feedback to dynamically alter the distortion introduced at the quantizer[6]. The simplest approach is to apply a scaling factor to the quantizer matrix to increase or decrease the magnitude of each element. However,

[6] The intermixture of intraframe and interframe coding also effects the bit rate but is usually set prior to encoding and not varied dynamically.

controlling bit rate reduces the coder's freedom to control quality. CBR video displays wider variations in visual quality compared to VBR video, which does not constrain bit rate.

3. Entropy Encoding

The quantized coefficients may be represented in a more efficient manner using source or entropy coding, thereby further increasing the compression gain. Video coders use a combination of run-length encoding and variable length coding.

Run-length encoding (RLE) is the simplest form of entropy coding and is frequently employed in both lossless and lossy compression schemes. Using RLE, a data block is parsed to locate sequences of repetitive values. Each sequence is replaced by a codeword consisting of a delimiter and the number of times the value is repeated. If the data block contains a great deal of repetitive information, a significant reduction in size is possible. Following quantization, the coefficient block typically contains a large number of zeros, especially amongst the high-frequency coefficients [6]. As the compression gain depends on the length of the sequence, rearranging the coefficient block as a vector in zig-zag fashion, starting from the DC coefficient down to the $F(8,8)$ coefficient, has been demonstrated to increase the run-length of the zeros. Different codewords are used, but the most common scheme consists of the run-length of zeros followed by the size or magnitude of next non-zero value. If no non-zero values remain, a special end-of-block codeword replaces the sequence.

After RLE, the quantized coefficient block is represented by a set of codewords with each representing a symbol drawn from a larger source alphabet. Variable-length coding (VLC) minimizes the average codeword length by assigning shorter codewords to the most probable symbols and longer codewords to the least likely symbols, and each codeword is uniquely decipherable. Huffman coding is the most widely used entropy-encoding algorithm and is guaranteed to produce a minimum average length, uniquely decipherable code [5]. The Huffman algorithm uses each symbol's probability of occurrence and builds a prefix code using an optimum binary-branching tree. Since both the coder and the decoder need to use the same codebook and generating a Huffman table

is computationally expensive, standard tables are normally pre-defined using data drawn from test images. An optimal representation is not guaranteed, but encoding and decoding are faster and the need to transmit the VLC table is avoided.

4. Quality of Reproduced Video

Given that video coders trade compression gain for image quality, quantifying the level of distortion introduced due to coding is useful in evaluating different coding schemes. A useful measure of image distortion D is to calculate the mean square error (MSE) between the original (x) and reconstructed ($\hat{x}$) images [6]:

$$D = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(x_{n,m} - \hat{x}_{n,m}\right)^2 . \qquad \text{(III.5)}$$

Using the MSE to quantify distortion D, the signal-to-ratio (SNR) is determined as

$$SNR = 10\log_{10}\frac{\sigma^2}{D}, \qquad \text{(III.6)}$$

where σ^2 is the input variance. The most widely published measure of image quality is the peak signal-to-noise ratio given by [6]

$$pSNR = 10\log_{10}\frac{K^2}{D}, \qquad \text{(III.7)}$$

where K is the maximum peak-to-peak value in the image, 255 for the typical 8-bit image. For example, a typical peak SNR for a typical JPEG encoded grayscale image is 28 dB at 0.5 bits/pixel [6].

Using MSE as a measure of image quality does have drawbacks. MSE does not distinctly relate to perceptual quality since all errors are given equal weight. Two compression techniques yielding the same MSE for an image may deliver slight differences in perceptual quality [6].

D. INTERFRAME CODING

Interframe coding exploits frame to frame correlation or temporal redundancy to deliver greater compression gains for a given level of quality. The degree of redundancy depends on the scene's motion content due to either motion of objects within the scene or

scene movement caused by a camera pan. Static scenes with little motion show a high amount of frame-to-frame redundancy. For example, the VTC scenario considered in this work assumes motion video sequences consisting of a "talking head," i.e, a single speaker talking against a static background. An opposite example is a scene change, where successive frames have completely different content.

Several source-coding techniques are employed to remove temporal redundancy including block updating, differential pulse code modulation (DPCM), and motion compensation. Each technique is suitable for a certain range of motion content. Generally, exploiting redundancy as motion content increases requires more complex techniques, which in turn decrease decoder robustness. As stated above, interframe coding offers the potential for a lower bit rate for a given level of quality. Conversely, interframe coding offers better quality for a given bit rate. The relative gain, as compared to intracoding, for the intercoding techniques presented here is documented in [53] for low and high motion video sequences.

1. Block Updating

The simplest interframe coding approach is a simple variation of intraframe coding. In low motion video scenes, such as "talking head" video, motion is confined to a small region within the scene while the background remains static. Block updating conserves bandwidth by coding and transmitting only those blocks that have changed perceptibly since the last frame [54]. Each block $f(i,j)$ is compared to its counterpart in the previous frame, and a distance metric is calculated. If the distance is below a certain threshold, no update for that block is transmitted. Otherwise, the block is intracoded as in Figure III.2 and transmitted. Block updating is sometimes combined with an aging scheme that periodically forces block updates, which mitigates hysteresis problems and guarantees that members joining a dynamic VTC session to receive the full scene within some set interval [45].

2. Differential Pulse Code Modulation

Another approach suitable for low motion video is DPCM. DPCM is a first order predictor that uses only the most recent sample to predict the next sample. Denoting the current frame as k and the reference frame as k - 1, DPCM subtracts the reference block $f(i,j,k$ - 1) from the predicted block $f(i,j,k)$. The resulting error block $e(i,j,k)$ represents the prediction error between the predicted block and the reference block. Although little correlation is left in the error block on average [48], the error block is compressed as shown in Figure III.2, which results in an approach known as hybrid video coding. If the prediction error is small, the dynamic range of the pixels is considerably reduced, possibly down to zero, and DCT-based coding yields a higher compression relative to intracoding the original block since the error block has a predominant lowpass characteristic.

Open loop DCPM has the disadvantage that errors introduced by quantization tend to accumulate over time at the decoder. Adding a feedback loop to the coder mitigates this problem. The predicted block is compared to a reconstructed version of the last frame maintained by the coder instead of the actual frame. Using the decoded frame as a reference compensates for quantizer error introduced by the coding process.

3. Forward Motion-Compensated Prediction

DPCM gives the best results when a scene is mostly static. With increasing motion content, the probability of poor correlation between the predicted block and the reference block increases. Past some point, DPCM actually yields inferior performance relative to intracoding. Assume that the predicted block contains a discrete object, such as a ball. If the ball does not move, DPCM gives good results since the best reference block is at the same coordinate as the predicted block. If the ball is moving, the best matching reference block is offset relative to the predicted block, and DPCM delivers poor results.

Motion compensation improves DPCM by comparing the predicted block to some region within the reference frame and finding a reference block that best matches the predicted block. The best match is determined by some criterion such as minimum

distance or maximum correlation. Since the search process is computationally intensive, real-time applications confine the search only to a small region about the predicted block while off-line coding may search the entire reference frame. The resulting error block is encoded as previously described under DPCM. Since the decoder needs the location of the reference block, a motion vector accompanies the encoded error block. The motion vector represents the location of the reference block as an offset (x,y) from the predicted block. DPCM is a special case of forward motion-compensation, using a motion vector of $(0,0)$.

Motion vectors add additional overhead to the encoding process with two implications for video coding. First, intraframe coders apply motion compensation at the macroblock level by associating four blocks with a single motion vector to reduce overhead. Second, motion compensation is only employed when a net gain in compression is possible over DPCM or intracoding after taking the overhead due to the motion vector into account. Most coders use the distance metric to determine the most appropriate method for encoding each macroblock, i.e., intercoding, either with motion compensation or DPCM, or intracoding.[7]

4. Bi-directional Motion Compensation

Forward motion compensation fails when no suitable reference exists in the previous frame. Such a situation arises whenever a scene change occurs or when motion reveals objects that are concealed in the previous frame. Bi-directional motion compensation improves coding in these situations by selecting the best reference block from either the previous frame or the subsequent frame. As before, the error block is encoded and transmitted along with a motion vector and a flag indicating which frame serves as the reference. The coder may also interpolate from the best matches in each reference frame although this approach requires transmission of two motion vectors.

The cost of adding bi-directional prediction is considerable and limits its suitability to off-line or non-real-time compression. The need to search two reference

[7] The picture type may further influence the decision process as in MPEG.

frames doubles both computational expense and buffer requirements. Also, the reliance on past and future frames requires that both the coder and decoder delay compression of the current frame until the subsequent frame is available.

5. Distance Metrics

In motion compensation, distance metrics are used to quantify the distortion between a candidate reference block and the predicted block. The best matching reference block generates the least distortion and thus provides the best match. Three distance metrics commonly employed are [6] [45]: mean squared error (MSE), sum of absolute differences (SAD), and absolute sum of differences (ASD). The corresponding mathematical expressions are given by:

$$e_{MSE} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(x_{m,n} - x^{R}_{m+i,n+j}\right)^2, \qquad \text{(III-8)}$$

$$e_{SAD} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left|x_{m,n} - x^{R}_{m+i,n+j}\right|, \qquad \text{(III-9)}$$

$$e_{ASD} = \frac{1}{MN}\left|\sum_{m=1}^{M}\sum_{n=1}^{N}\left(x_{m,n} - x^{R}_{m+i,n+j}\right)\right|, \qquad \text{(III-10)}$$

where $x_{n,m}$ represents the pixel intensities within the predicted block while $x^{R}_{m+i,n+j}$ represents the pixel intensities in the, possibly offset, reference block. The reference block is offset relative to the predicted block by the motion vector (i,j).

Although several H.261 video codec implementations employ MSE as a distance measure [6], MSE requires expensive multiplication operations, which makes it less suitable for real-time applications. SAD and ASD require the less complex absolute value operator and otherwise require only addition operations. SAD was incorporated into the H.263 test model [55], an approach probably adopted by commercial implementations. ASD has found use in block updating since taking the absolute value after the summation reduces the impact of noise introduced during video capture, thereby reducing spurious background updates in low motion video [45].

6. Hybrid Video Coding

Hybrid video coding combines motion compensation with the DCT-based coder shown in Figure III.2. A functional block diagram of a hybrid coder is shown in Figure III.5. Similar to intracoding, the current frame is broken into a sequence of macroblocks, and a separate coding decision is made for each macroblock. The motion estimation block compares each macroblock to the reference frame(s) and decides whether intracoding or intercoding is more appropriate. For example, Telenor's H.263 test model [55] employs a SAD-based coding decision algorithm. If intracoding is indicated, DCT-based compression is applied to each individual block within the macroblock. If intercoding is selected, the reference macroblock is subtracted from the predicted macroblock, and the error block is encoded. The motion vector is encoded separately using a VLC although motion vectors are optional for simple DPCM.

Figure III.5 also illustrates the feedback path used to prevent the accumulation of quantization errors at the decoder. After each macroblock is quantized, the quantization and transform operations are reversed, and the results are used to update the reference frame. Not shown is the controller functionality. The controller implements either open-loop or closed loop rate-control, in coordination with the network, by controlling distortion introduced in the quantizer and by controlling encoding decisions available to the motion estimation block.

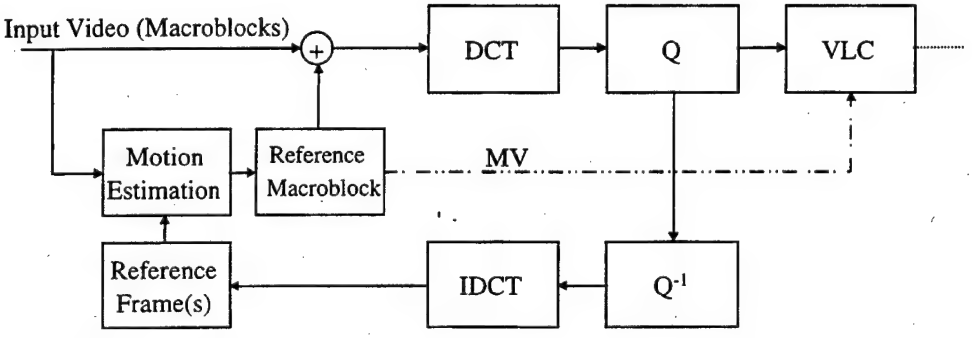

Figure III.5: Hybrid Video Coder with Motion Compensation and DCT-based Compression.

E. ERROR ROBUSTNESS

Transmission errors are an inevitable part of any communication network and occur both within the channel and within the network. Communication channels are characterized by bit error rate (BER), typically 10^{-9} for fiber-optic systems and considerably more for copper-based wireline and wireless systems. Random bit errors or burst errors due to channel noise may corrupt either the payload or the packet header. Packet header errors are the more serious of the two, raising the potential for misrouted packets or preventing the network from identifying the packet. Losses may be mitigated with forward error correction and retransmissions, but the latter approach is untenable with real-time traffic. ATM networks only check for errors in the cell header and are able to correct single-bit errors [18]. If multiple bit errors are detected, the cell is discarded. The AAL layer at the receiver may handle payload bit errors or leave error handling to higher layers. Network losses occur due to buffer overruns at network nodes during periods of congestion or when the arriving aggregate traffic prevents the switch from servicing each connection to its required QoS. Although network architectures, such as ATM, allow a call to specify cell loss probability prior to call acceptance, cell losses do occur, especially if the transmission path employs a wireless interface. The impact of transmission errors depends of the error resilience of the codec.

Each cell loss or bit error degrades the quality of the reconstructed video stream through two mechanisms depending on the type of video coding employed. Assume that a transmission error occurs such that a single macroblock is decoded incorrectly. The immediate impact is spatial corruption within the current frame [7]. Since the error disrupts the decoder's synchronization with the bit stream, the corruption spreads spatially in scanline fashion until the decoder locates a valid symbol for resynchronization. Therefore, the visual corruption usually spreads through the remainder of the parent GOB or to the end of the frame.

With intraframe coding, spatial errors do not persist beyond the affected frame since each frame is coded independently. Interframe coding, while giving greater compression gains, increases the impact of spatial errors by providing a propagation path

through subsequent frames. Again, consider the presence of one or more corrupted blocks in the last decoded frame. In interframe coding, the last decoded frame serves as a reference for predictive coding. Any error block received in the current frame that references a corrupted block yields another corrupted block. Therefore, spatial corruption propagates temporally. With motion compensation enabled, scene motion *carries* decoding errors spatially through the scene. This is particularly distracting since the human eye tends to follow motion [7]. Duration of temporal errors is dictated by the rate at which intracoded macroblocks are transmitted, which is in turn dictated by the codec. Factors impacting the relative error resilience of several popular codecs are presented below.

1. Motion JPEG

Motion JPEG treats the video stream as a sequence of still images, compressing each frame using JPEG. Since each frame is encoded independently, decoding errors are limited to the duration of the affected frame.

2. MPEG

MPEG-1 and MPEG-2 are designed to deliver high-quality audio-video compression for applications, such as CD-ROM multimedia, broadcast digital video, and high definition TV. MPEG employs the GOP format shown in Figure III.1 to provide a tradeoff between compression gain and random access within the video stream [6]. A GOP includes three picture types; each picture type limits the allowable macroblock types. I- and B-pictures are anchor pictures and serve as reference frames. I-pictures allow only intracoded macroblocks. P-pictures allow intracoding and forward motion prediction from the last anchor picture. B-pictures allow intracoding, bi-directional motion prediction, and interpolation and use the last and next anchor frames as references. Although not specified by the MPEG standard, an *N*- picture GOP normally starts with an I-picture followed P-pictures every *M* frames. The remaining frames are encoded as B-pictures as shown in Figure III.6. A greater value of *N* offers greater

compression gain at the expense of random access since the decoder must start at an I-picture.

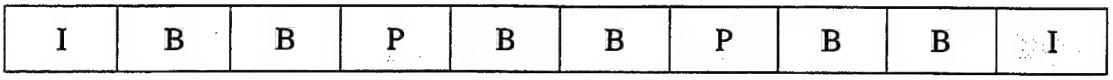

Figure III.6: Typical GOP, $N = 9$, $M = 3$.

If an error occurs in any anchor picture, errors may propagate through the remaining pictures in the GOP until the next I-picture is received. An I-picture decode error is the worst case and results in the longest propagation cycle. Since MPEG employs motion compensation, decoding errors propagate spatially as well as temporally and have been observed to grow and shrink depending on motion within the frame.

3. H.263

The ITU standard H.263 defines a low-bit-rate video codec for video transmission over the PTSN using V.34 modems. H.263 is optimized for bit-rates of 28.8 kbps and less and offers quality superior to MPEG at bit-rates less than 64 kbps.

H.263 employs the video hierarchy shown in Figure III.1 without the GOP structure. H.263 coding resembles the concept of MPEG P-pictures. All coding decisions are made at the macroblock level and each macroblock is either intracoded or intercoded using forward motion compensation. To bound error propagation, the standard specifies that a macroblock must be intracoded at least once every 132 frames [56]. The lack of the equivalent of an I-picture to reset every macroblock at once, while deliberate, leaves H.263 vulnerable to prolonged error propagation. Even with the mandatory spacing of intracoded blocks, some types of motion lead to almost indefinite error propagation [8].

4. Error Propagation

To place error resilience in context, consider the worst-case error propagation using M-JPEG, MPEG and H.263 compression under the scenario summarized in Table I.1. With M-JPEG, an error in one frame is corrected upon receipt of the next frame.

The robust nature of M-JPEG makes it suitable for broadband video conferencing [9]. Error propagation in MPEG depends on the GOP size. A typical reported GOP size is twenty pictures and, given that an error occurs in the I-picture, the worst-case propagation is twenty frames. For an H.263 coded stream, the worst-case error propagation depends on how often individual macroblocks are intracoded. The H.263 standard specifies a maximum limit of 132 frames between updates [56]. Assuming an error occurs in an intracoded block and the block is not intracoded again for 132 frames, the error could persist as long as 132 frames and possibly even longer given the right motion patterns [8]. Table III.1 summarizes the worst-case error duration for each of the three codecs for a frame rate of 10 fps.

Coding Scheme	Worst-case error propagation (seconds)
JPEG	0.10
H.263	13.20
MPEG	2.00

Table III.1: Error Propagation in Popular Video Codecs.

F. SUBBAND AND WAVELET CODING

Subband and wavelet coding are additional techniques for compressing still images and have been shown to offer slightly better image quality than DCT-based schemes for similar levels of compression at the cost of greater computational complexity [50]. Subband and wavelet coding are fundamentally similar in that both decompose the image into regions representing different bands of spatial frequencies present in the image. Subband coders apply a series of filters to the image and then decimate the resulting bands to avoid oversampling while wavelet coders perform filtering and decimation simultaneously [48]. Of the two methods, wavelet techniques are more common and are examined further here.

In contrast to the DCT, a discrete wavelet transform (DWT) filters and decimates an image into regions containing mixtures of the high and low frequency details within the image. Decomposition is performed using two analysis filters. The first extracts low-frequency content, the signal average, and the other extracts high-frequency content, the signal details. Example analysis filters for a four-tap biorthogonal DWT are given by [48]:

$$H_0(z) = -1 + 3z^{-1} + 3z^{-2} - z^{-3} \tag{III-11}$$

$$H_1(z) = -1 + 3z^{-1} - 3z^{-2} + z^{-3}. \tag{III-12}$$

The inverse transform is performed using the following synthesis filters:

$$G_0(z) = \left(1 + 3z^{-1} + 3z^{-2} + z^{-3}\right)/16 \tag{III-13}$$

$$G_1(z) = \left(-1 - 3z^{-1} + 3z^{-2} + z^{-3}\right)/16. \tag{III-14}$$

Image compression proceeds as shown in Figure III.7. A first order decomposition creates four 2-D subbands from the original image. Each subband results from the appropriate application of the analysis filters in the horizontal and vertical directions and decimation by a factor of two. For example, applying Eq. (III.11) in both the horizontal and vertical directions generates the LL band. Applying Eq. (III.11) in the horizontal direction and (III.12) in the vertical direction results in the HL subband. The remaining subbands are obtained in a similar manner. Each subband captures certain image features. The LL subband retains the low-pass information within the image and displays a coarse representation of the original image. Since most images have a low-pass characteristic, most of the image's energy is found in the LL subband. High-frequency information results from edges, which provide visual cues for image recognition. The HL and LH subbands contain vertical and horizontal edge information, respectively, while the HH subband contains diagonal edge information.

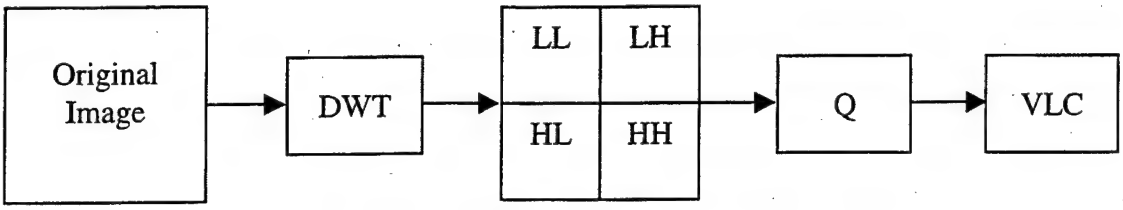

Figure III.7: DWT-based Image Compression.

The wavelet transform is invertable and lossless and, like the DCT, produces no compression gain. The compression gain results from quantization and entropy coding of the wavelet coefficients. As with the DCT, the higher frequency coefficients tend to be less significant, so most of the compression gain is realized from compacting the detail subbands, especially the HH subband. In the layered coder proposed by McCanne and Vetterli, the HH subband is discarded entirely [45]. Subbands are usually quantized independently. The LL band behaves much like the original image and can be compressed using traditional transform-based techniques such as JPEG [57]. The remaining subbands are uniformly quantized using a stepsize proportional to the variance of the coefficients in that subband [48]. Since the higher subbands tend to have a large number of zeros following quantization, run-length encoding and entropy encoding significantly increase compression. Zig-zag reordering provides no advantage in the upper bands, so RLE occurs scanline fashion, either horizontally or vertically. Alternatively, the quantized coefficients are grouped and vector Huffman encoded [58].

Greater compression is possible by further decomposing the image. Figure III.8 displays a second-order octave-band decomposition obtained by applying the analysis filters to the LL subband as described above. A higher-order decomposition is generated by repeatedly decomposing the lowpass subband. The lowpass band is quantized using transform-based techniques while the remaining subbands are quantized as described above. The increase in the number of bands allows quantization and encoding to be further tailored to emphasize perceptual details over less perceptible background noise. Alternatively, the interdependencies among the subbands can be exploited using zero-tree entropy coding [59]. Zero-tree coding is analogous to zig-zag scanning in DCT-based

compression. The tree grows from a single coefficient in each of the low frequency bands and gathers coefficients in higher frequency bands that correspond to the same spatial location in the original image. Each additional subband increases the size of the tree by a power of four. Zero-tree encoding combines elegantly with bit-allocation since encoding may stop once the target bitrate is met. Conversely, the decoder may stop once a desired level of quality is achieved.

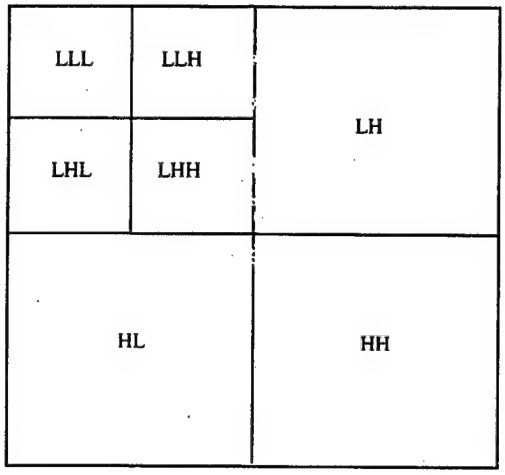

Figure III.8: Octave-band Decomposition.

Wavelet-based compression schemes offer some advantages over DCT-based schemes. The DCT-based approaches achieve compression gain by removing high-frequency content from the image by zeroing the high-frequency coefficients during quantization. Wavelet transforms separate the image into regions of high and low frequency content, thus allowing more efficient bit allocation since different regions may be quantized and coded differently. This is advantageous since the DWT coder has the option of preserving more or less edge detail to improve perceptual image quality at comparable pSNR to the DCT. Another advantage is that wavelet transforms are not applied to blocks within the image but are instead applied to the entire image. Therefore, at low pSNR, while the DCT demonstrates blocking artifacts wavelet transforms typically display a more visually pleasing smoothing effect. In general, wavelet transform coders offer compression gains, at comparative pSNR, superior to DCT-based coders. When

comparing the state-of-the-art coders, wavelet-based coders offer 1 dB improvement in pSNR over DCT-based coders [50].

Several drawbacks relative to DCT-based compression have limited the utility of wavelet-based video compression. Wavelets achieve quality superior to DCT-methods by processing the entire image or frame. Motion-compensated video coding exploits temporal correlation at the macroblock level. Although the error block could be transformed via a DWT, no significant advantage has been determined over the DCT, and the computational effort is greater [50]. Many software and hardware "fast" implementations of the DCT require less than one multiplication per coefficient. Wavelet transforms are usually bounded to at least one multiplication per coefficient.[8]

This chapter presented the tools required for compressing motion video: transform methods, quantization, and entropy coding. These tools can be applied to individual frames independently as in intraframe coding, or used in conjunction with prediction schemes that capture frame-to-frame correlation as in interframe coding. An important consideration is that the choice of methods impacts both the complexity and error robustness of the coder. Therefore, codec suitability for a particular application is to some degree dependent on the host networking environment. Wavelet-based coding allows flexibility with frequency content selection to improve compression. The frequency decomposition offered by DWTs also provides a powerful tool for devising more robust schemes for video transmission as detailed in the next chapter.

[8] The fast Haar transform is the exception, which requires no multiplication operations [60].

IV. LOW-COMPLEXITY LAYERED VIDEO CODING

Current coding standards, such as H.263 and MPEG, make no explicit allowance for network transmission and are severely degraded by both bit errors and packet losses [7]. Packet losses are preventable to some extent with proper QoS guarantees, but losses due to congestion still occur. Of further concern is the fact that tactical wireless links exhibit much higher BERs relative to wireline connections. Putting aside the matter of BER as outside the control of network applications, most approaches to reducing the impact of congestion involve feedback-based rate-control schemes that change the coder's quantization, resolution, or frame rate. As discussed in Chapter II, RTP provides a framework for a multimedia application to gauge the level of congestion within the network via receiver reports and vary its target bit rate accordingly.

A second drawback is the poor flexibility exhibited by traditional video codecs in multicast scenarios when video is transmitted over heterogeneous, packet-based networks. These codecs transmit the video signal as a single stream of packets. The combination of a single video stream and a heterogeneous network suffer from many limitations [12]. Consider the problem of delivering video to a multicast group consisting of several recipients connected over the heterogeneous network shown in Figure IV.1. Examining the transmission paths leading from the sender to the different recipients reveals an obvious stratification in available bandwidth[9]. In this scenario, the sender faces a dilemma when selecting an appropriate encoder quality. Transmitting high quality, high bandwidth video is both acceptable and desirable for some recipients. However, low bandwidth recipients will experience high packet loss with a commensurate degradation in received video quality. Supporting the lowest common denominator forces all recipients to view lower quality video, thereby underutilizing high bandwidth links and leaving those recipients dissatisfied.

[9] A similar heterogeneity could exist in each user's processing and display capabilities.

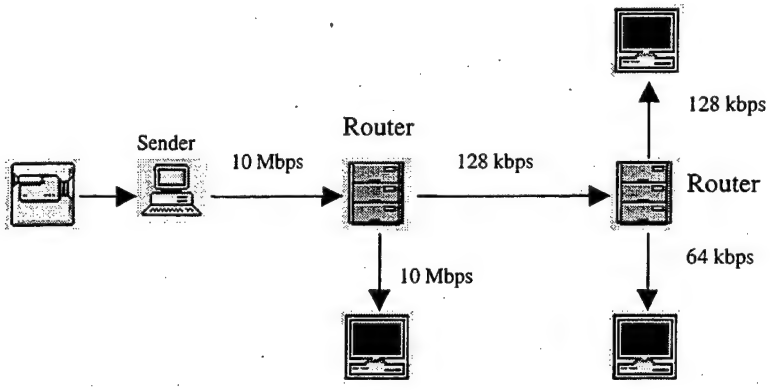

Figure IV.1: Video Transmission over a Heterogeneous Network from [45].

This chapter addresses these concerns by considering a layered video coder that is more suitable for network transmission. The concept of layered coding, especially in the context of receiver-based layered multicast (RLM), and previous layered coder proposals are examined. The chapter's primary focus is on a new SNR-scalable layered coding scheme appropriate for tactical applications with emphasis on robust transmission and low complexity. Error robustness is provided by eschewing motion prediction in favor of macroblock updating, which significantly limits the temporal duration of decode errors and eliminates any spatial migration. Layering is accomplished via the fast Haar transform (FHT) with the exact layering structure tailored to video content. The VTC session is assumed to consist of both low-motion video, such as a "talking head", and static displays, such as slide presentations. Handling both types of content with a single layering scheme requires unacceptable compromises since the frequency characteristics of each are different. Therefore, the coder is optimized to handle each type of content separately by including separate layering structures and custom VLC tables. Finally, the rate control problem is examined, and an approach is proposed to reduce a k-dimensional rate-control problem to a simple 1-D table lookup.

A. BACKGROUND

Several approaches are available to meet the diverse quality expectations in the multicast group. The sender could encode the input video as a series of separate streams,

where each stream targets a different quality level and target bit rate. Each stream is then transmitted to a different multicast group. Recipients then subscribe to that multicast group offering the desired quality and bit rate. A multicast group such as that shown Figure IV.1 would potentially require targeting three different bandwidths. However, separate encoding presents some liabilities [45]. Transmitting several streams duplicates content and requires far more bandwidth. Encoding several streams simultaneously requires considerably more computational effort than a single stream and limits this approach primarily to non-interactive video-on-demand applications. Another approach is to use transcoding at routers wherein a high-quality video stream is decoded and then encoded to a lower quality for further transmission on a lower bandwidth network [45]. However, transcoding requires specialized hardware in the transmission path, and the additional delay introduced in reprocessing the video stream makes it less suitable for interactive applications.

As discussed above, feedback messages allow the sender to estimate network conditions and adapt to the onset of congestion, thereby reducing the load on the network and ensuring that all recipients receive a minimal level of quality. RTP provides a mechanism for receiver reports but leaves the actual mechanism for interpreting reports and making changes to the application. Other schemes have been developed mainly for use over LANs but could be adapted for multicast applications hosted over an ATM network. One scheme proposed by Bolot and Turletti [61] employs negative acknowledgements to indicate network state when the number of recipients is ten or less and uses QoS messages sent periodically with some probability. Sakatani [62] uses collisions detected at the MAC level and round-trip delay to measure the effect of congestion. Once congestion has occurred, quantization and frame rate are dropped to a "slow start" bit rate. If indications of congestion disappear, the original bit rate is resumed. Other schemes have been proposed by [63]-[65].

However, heterogeneous networks complicate application of feedback-based rate-control schemes. In a multicast environment, each recipient in a VTC may observe different degrees of congestion. The sender's task of interpreting the network state and

making appropriate changes is greatly complicated when sender reports indicate that congestion affects only a small subset of the multicast group. Aggressive response lowers quality to the entire multicast group while a more conservative response tacitly drops some recipients, at least temporarily. Feedback-based control in general is problematic. With high-bandwidth networks, rate-control schemes may not respond fast enough to be beneficial. In low-bandwidth networks, any feedback scheme consumes bandwidth although most attempt some form of conservation. For example, RTP scales the receiver report rate to the size of the multicast group. Still, the notion of rate control leads back to the issue that selecting a single level of video quality in a heterogeneous environment is problematic.

Layered video coding, especially in the framework of receiver-based layered multicast (RLM) [45], provides a solution to the shortcomings outlined above. A layered-video coder encodes the video stream as a base layer and a series of enhancement layers, arranged in a hierarchical fashion. The base layer provides a minimum acceptable level of quality while the enhancement layers progressively refine the quality of the received video sequence.

Layered video coding with RLM offers greater flexibility in handling the video stream by moving bandwidth management from the sender to the network and the individual recipients. The sender generates a layered video stream at the highest quality (bandwidth) supported by the network to which it is directly attached. Each member of the multicast group then subscribes to some or all of the layers. The exact number depends on available bandwidth and the video quality desired. If high packet losses are experienced, the recipient drops layers until satisfactory reception is obtained. Within the network, the video stream traverses a heterogeneous mixture of subnets. Each subnet carries the maximum number of layers within the bandwidth available, retaining the most perceptually important layers and dropping the rest. Figure IV.2 shows this approach using the heterogeneous network portrayed in Figure IV.1. Transmitting the video stream as a series of scalable layers maximizes utilization of each link and maximizes the video quality available to each recipient.

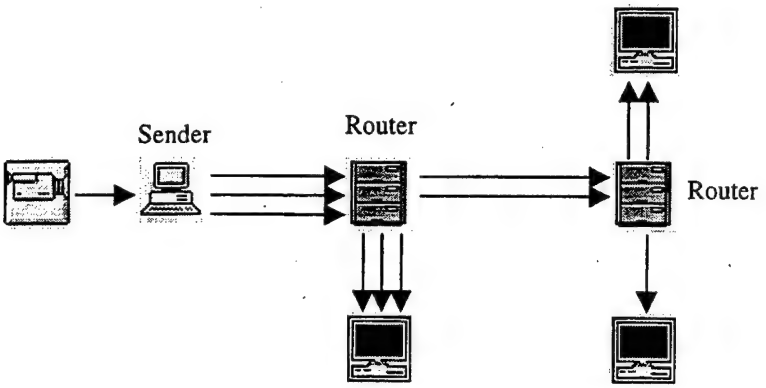

Figure IV.2: Video Transmission Using RLM.

RLM as originally described by McCannes et al. [45] implicitly provides congestion control without feedback via recipient subscriptions. When experiencing high-packet loss, recipients have the option of dropping the less important layers. As layers are dropped, routers stop forwarding their packets, thus preserving bandwidth for more perceptually important layers. This allows more graceful degradation in video quality in the presence of both congestion and other changes in network loading. The sender does not play an active role in congestion control although receiver reports could be used to drop or manipulate the upper layers. RLM can be improved by providing QoS guarantees for each layer and exploiting the hierarchical nature of layered video in network scheduling decisions. Chapter II discussed methods for multicast transmission of layered video with QoS guarantees using ATM; scheduling algorithms for layered video are covered in Chapter VI.

RLM also does not explicitly increase error resilience except each subnet carries only those layers capable of being transmitted without excessive packet losses. However, research [13] indicates that layered video provides more error resilience than a single video stream of similar bandwidth. Spreading errors across multiple layers means that fewer errors occur in the base layer relative to a single stream, and errors in the enhancement streams are less noticeable. With ATM networking, QoS can be negotiated asymmetrically to ensure that fewer errors occur in the most important layers.

B. LAYERED VIDEO CODING

Delivering layered, scalable video involves considerations in addition to those covered in the last chapter for traditional coders. The primary concern is effectively separating the video stream into hierarchical layers as shown in Figure IV.3. The video stream consists of a base layer that offers acceptable quality and a series of enhancement layers that progressively improve quality in terms of pSNR, frame rate, or resolution. An effective layering scheme creates layers that provide gradual but perceptible increases in video quality. Transmitting an additional layer that does not improve quality merely wastes bandwidth. An effective layering scheme should also create the layering hierarchy without significantly increasing computational expense as compared to encoding a single stream and with minimal additional bitstream overhead.

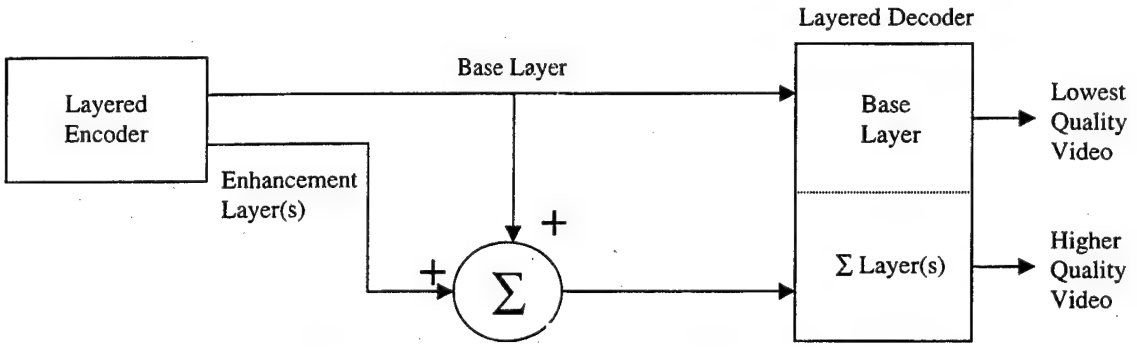

Figure IV.3: Overview of Layered Video Coding/Decoding.

Next, we consider some basic approaches for implementing the layering operation implied in Figure IV.3. Two avenues are considered. First, progressive image refinement schemes, such progressive JPEG and pyramid coding, easily extend to layered coding. Second, as mentioned in Section III.F, multiresolution techniques employing subband/wavelet image coding extend in a natural fashion to layered coding. Each of these techniques is explored and illustrated with past and current research on layered coder design.

1. Progressive JPEG Encoding

Progressive encoding is one of the four encoding modes defined in the JPEG standard and represents an extension to the baseline sequential coder presented in Figure III.2 [66]. Progressive JPEG prepares the image for encoding in the same manner. The image is broken into 8×8 blocks, transformed with the 2-D DCT, and quantized using either JPEG standard or customized tables. The difference lies in the manner in which the quantized DCT coefficients are encoded. Progressive coders segment the DCT coefficients and encode them in multiple passes with each pass containing a subset of the frequency content. The goal is to first transmit the most perceptually important frequency content and then progressively improve quality with the remaining passes. Segmentation is performed via spectral selection or successive approximation.

From Figure III.3, the DCT coefficients are arranged from low frequency components in the upper left corner to high frequency components in the lower right corner. Spectral selection segments DCT coefficients into spectral bands for encoding, where each band includes a discrete set of spatial frequencies. The first spectral band includes the DC coefficient and some number of neighboring AC coefficients. Successive bands incorporate higher frequency coefficients until all coefficients have been selected. There are various ways to select the spectral bands. One method is to treat each diagonal, starting with the DC coefficient and working right and down, as a separate spectral band. Another method is to group coefficients with similar variances, where each coefficient's variances is calculated using representative test images [6].

Spectral selection tends to produce blocking artifacts when using only a few spectral bands since low frequency content is transmitted first. Successive approximation provides more visually pleasing performance by transmitting a portion of all non-zero DCT coefficients in each pass [6]. Each coefficient is essentially a binary value and, within that binary value, the most perceptible content is carried in the most significant bits. Therefore, on the first pass, a specified number of the most significant bits for each non-zero coefficient are encoded. On successive passes, the less significant bits are encoded. Successive approximation yields a more graceful transition in image quality

than spectral selection since each pass includes some high frequency content. However, successive approximation incurs greater coder complexity compared to spectral selection [6].

Progressive JPEG may be viewed as providing a "preview" image and then successively decreasing the distortion by transmitting additional coefficients. A similar approach in layered coding is to transmit a base layer and then an enhancement that mitigates errors in the base layers.

Rhee and Gibson [13] have proposed a two-layer coding scheme targeting ISDN, enabling support for one or both B channels dependent on the available capacity (64–128 kbps). One channel transmits an H.261 encoded base layer while the other channel sends an enhancement layer constrained to no more than 64 kbps. As H.261 is similar to the H.263 codec described in Section III.E, only the enhancement layer is covered here.

After encoding a frame, an H.261 coder decodes the frame to serve as a local reference for motion compensation when encoding the next frame [67]. Rhee and Gibson's proposed coding scheme compares the original frame to the decoded frame and determines the MSE introduced by coding for each block. The block errors are sorted from highest to lowest, and the B blocks with the highest error are selected for enhancement. While the number of blocks selected is fixed (160 in the simulations), the location of the blocks varies each frame depending on scene content. After the blocks are selected, b bits are allocated to each block such that Bb equals the desired bit rate per frame. The bits are allocated to encode the error at each pixel within a selected block based on a bit allocation scheme that considers the observed error variance at each pixel in test video sequences. Pixels demonstrating larger error variances are allocated a greater proportion of the bits; the bit assignment remains constant throughout the video session.

Another proposed layered refinement scheme based on H.261 from Rhee and Gibson [68] uses the refinement layer to more accurately describe motion present within the frame. H.261 performs motion compensation at the 16×16 macroblock level, which sacrifices the more precise motion information available using 8×8 blocks but is faster

computationally [67]. The enhancement layer considers the displacement of the individual blocks comprising a macroblock and yields more accurate motion prediction and better visual quality[10].

The baseline H.261 coder performs macroblock level motion prediction by comparing the current 16×16 macroblock to every macroblock in the previous frame and selecting the best match. The difference between the macroblocks is quantized, encoded, and stored along with the macroblock motion vector. In a parallel operation, block-level motion prediction is performed for the four blocks comprising the current macroblock. The macroblock motion vector is subtracted from each of the individual block motion vectors, giving four residual motion vectors. The residual motion vectors are stored along with their respective encoded difference blocks in the refinement layer. At the decoder, both the baseline H.261 and refinement streams are decoded simultaneously. Within the H.261 stream, the macroblock motion vectors and associated difference macroblocks are used to update the current frame. If the refinement layer contains information for a particular macroblock, the baseline-decoded blocks are replaced with updated blocks using the block-level motion vectors.

2. Pyramid Coding

The pyramid coding scheme proposed by Burt and Adelson [69] extends well to a layered representation of still images and has been extended into the temporal domain for video coding [48]. Pyramid coding employs a simple but effective prediction scheme. The image is low-pass filtered, decimated by a factor of two, and then quantized. The result is a base image that is a coarse representation of the original. Next, the base image is interpolated back to the original image's resolution, filtered, and subtracted from the original image to produce a prediction error. If the image has a low frequency characteristic, usually a good assumption, the error image is highly correlated and compresses very well. The base image is stored or transmitted using lossless compression while the error image is compressed using a lossy coder. At the decoder,

[10] H.263 offers block-level motion compensation as an option [56].

the error image is added to an interpolated version of the base image to reconstruct the original image. Although pyramid coding is lossy, the error results only from quantization of the error image, which may be bounded through proper choice of the quantizer.

The previous description applies to one-step pyramid coding. A multi-step pyramid is implemented by successively repeating the filtering and decimation operations until the desired size base image is produced; each step reduces the size of the image by a fourth. For an n-step pyramid, the result is a heavily filtered base image and a series of $n-1$ error images. The drawback to a multi-step pyramid is increased computational demand as well as increased encoding delay and increased over-sampling of the image.

The CafeMocha encoder [70] uses pyramid coding to form two layers, and each layer is transmitted to a separate multicast group using two RTP sessions. CafeMocha transmits video at a resolution of 320×240 with 4 bits/pixel. The base layer uses the popular CU-SeeMe video coder [1] at a lower resolution of 160×120, and the enhancement layer uses a pyramidal coder to improve the resolution to 320×240. The CU-SeeMe coding algorithm uses block replenishment followed by lossless compression. A 320×240 frame is first decimated to obtain a 160×120 base frame. Each 8×8 block in the base frame is then compared to its counterpart in the last base frame and is selected for transmission if the difference exceeds a threshold. The selected blocks are losslessly compressed and placed into packets of no greater than 1000 bytes to avoid fragmentation along the transmission path.

Instead of forming an error frame, the pyramid coder generates error blocks. Each 8×8 block selected for transmission in the base layer is interpolated to give a 16×16 macroblock. The interpolated macroblock is then subtracted from the corresponding macroblock in the 320×240 image to form an error macroblock. The difference block is losslessly compressed using run-length coding and packetized as above. The results in [70] indicate that the addition of a second layer improves visual quality compared to a 320×240 CU-SeeMe video stream when subjected to a 50% packet loss rate.

Gharavi and Partovi have proposed a multi-grade, layered coding scheme that combines elements of pyramid and subband coding along with DPCM [71]. Instead of providing increasing grades of quality at a fixed resolution, the coder provides scalable resolutions and accepts lower image quality at higher resolutions. Three layers are employed: a base layer (L1) and two contribution layers (C1 and C2). The different resolutions are obtained by combining the appropriate layers prior to the decoder as indicated in Table IV.1.

Quality Grade	Resolution	Layers Required
Q1	352×240	L1
Q2	704×480	L1+C1
Q3	1408×960	L1+C1+C2

Table IV.1: Resolutions Supported in Gharavi and Partovi's Layered Coder.

Video is captured at the highest resolution (Q3) and low-pass filtered and decimated to obtain the next lower grade (Q2), which is in turn low-pass filtered and decimated to obtain the lowest quality video (Q1). Q1 is encoded using a hybrid DCT/DPCM scheme compatible with H.261. The Q2 and Q3 video streams are encoded separately but in the same manner using hybrid subband/DPCM encoders.

3. Wavelet and Subband Coding

Wavelet and subband coding provide a good starting point for designing a layered coder since each image or frame is resolved into a series of subbands that follow a strict hierarchy [48]. As discussed in Section III.D, a two-level wavelet decomposition of an image yields an average subband LL, representing the low pass frequency components of the image, and the detail subbands LH, HL, and HH, representing higher frequency detail in the horizontal, vertical, and diagonal directions, respectively. The following is one of several approaches to realize a simple layered coder using a wavelet transform:

- Compress each frame separately by using the wavelet transform.
- Quantize and entropy encode each subband separately.

- Form three layers based on the frequency content: a base layer (LL subband), a first enhancement layer (LH and HL subbands), and a second enhancement layer (HH subband).

A coder employing this approach is shown in Figure IV.4. At the receiver, the layers are decoded and inverse wavelet transformed prior to video display. If any layers are dropped due to bandwidth (or possibly errors), those wavelet coefficients are assumed to be zero and the frame is reconstructed using the remaining detail subbands.

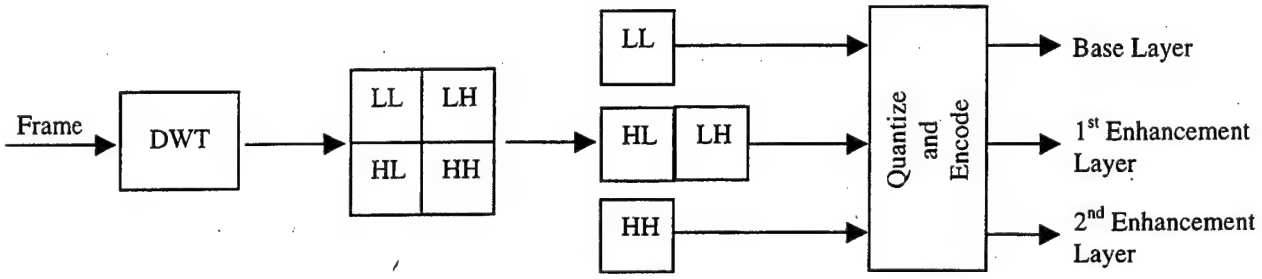

Figure IV.4: Basic Layered Video Coder Using Wavelets.

If more layers are desired, the process can be repeated at the coder by applying the wavelet transform to the average (LL) subband to generate four higher order subbands. Following the approach outlined above, the compressed video could be transmitted using as many as seven distinct layers.

Bahl and Hsu have proposed a wavelet-based layered coder incorporating content sensitive spatial decomposition and multiresolution coding [72]. Spatial decomposition is performed via a split-and-merge algorithm [73]. A frame is split into blocks of identical size and then adjacent blocks of similar variance are merged to generate regions of common perceptual importance. After applying the algorithm, the results are saved as a segmentation mask and reused for subsequent frames. A new segmentation mask is only calculated if significant motion occurs within the frame.

The coder decomposes each block using the fast Haar transform (FHT) and then applies motion compensation, quantization, and variable-length coding to each subband. Bit allocation is performed in proportion to the variance exhibited within each subband.

Transmission is prioritized by subband and region and, optionally, the receiver can request priority updates for regions corrupted by packet loss within the network.

McCannes et al. have performed the most extensive work on the problem of multi-cast video by proposing the RLM architecture for delivering multi-cast video over heterogeneous networks [12]. In a follow-on work, the authors break the multicast video problem into two areas, the compression problem and the transport problem, and propose a comprehensive solution for both problems [45]. The compression problem is met with their proposed hybrid DCT/wavelet layered codec. The codec provides robust error resilience, low coder complexity for good run-time performance, and acceptable compression performance.

Error resilience is provided through macroblock-based conditional replenishment wherein only the macroblocks that change in the current frame are encoded for transmission. While block replenishment does not offer the same compression gain available with motion compensation, the authors argue that the difference is negligible compared to improved quality when considering packet loss.

After blocks are selected for replenishment, they are compressed spatially using a hybrid DCT/wavelet scheme. Each 16×16 macroblock is decomposed into four subbands. The LL band is created using a 1/3/3/1 biorthogonal wavelet, and the remaining subbands are created using the discrete Haar transform [48]. The HH band contributes little energy to the reconstructed frame and is discarded. The LL block is further transformed with a DCT and the resulting coefficients are progressively encoded using spectral selection. The remaining LH/HL subbands are combined and are also progressively encoded using embedded zero-trees.

Once all selected blocks within the current frame are encoded, a spatio-temporal hierarchy is created combining spatial and temporal layering. Within each encoded block, the progressively encoded DCT and wavelet coefficients are organized into a number of spatial layers. The possible combinations of bit-rate between spatial and temporal layers is a two-dimensional region where every trajectory provides a compromise between visual quality and the rate of frame updates at increasing bit rates.

C. A LOW-COMPLEXITY ADAPTIVE LAYERED CODER DESIGN

In this section, we propose a new layered coder design. The goals in proposing a new coder are threefold. First, tactical considerations limit transmission bandwidth and place a premium on robust transmission. These considerations determine the type of compression techniques that are desirable or even feasible in a tactical video coder. Second, previously reported layered coding efforts are very diverse with emphasis on different network architectures or applications. Consensus on identifying a structured approach to designing layered coders or quantifying those parameters that make a layered coder *effective* is lacking. Third, a working coder provides a source for gathering statistical traffic data that is used in later chapters to model layered video traffic for network simulations and to examine error concealment issues. A working implementation of this coder is provided by [74] and was used to evolve the design.

The guidelines observed in designing the layered coder flow from both the tactical VTC application and the considerations for designing an effective layered coder. The application imposes the following requirements. First, the coder must adaptively optimize compression for both low motion video and static slides. Second, the coder must possess a low complexity architecture to minimize coding delays and power requirements. Third, the coder must provide error resilient decoding at high packet loss rates. Fourth, the coder must constrain the bit rate to a predetermined average. Finally, the coder must meet the performance specifications listed in Table I.1.

Implementing an effective coder within the above constraints involves due consideration of the following elements. First, the coder should transmit a base layer with acceptable quality and two (or more) enhancement layers such that each progressively improves *perceptual* quality. Second, the coder should minimize the bitstream overhead required to accommodate the layering structure.

A functional diagram of the proposed coder is shown in Figure IV.5. Details for each component are provided in subsequent sections.

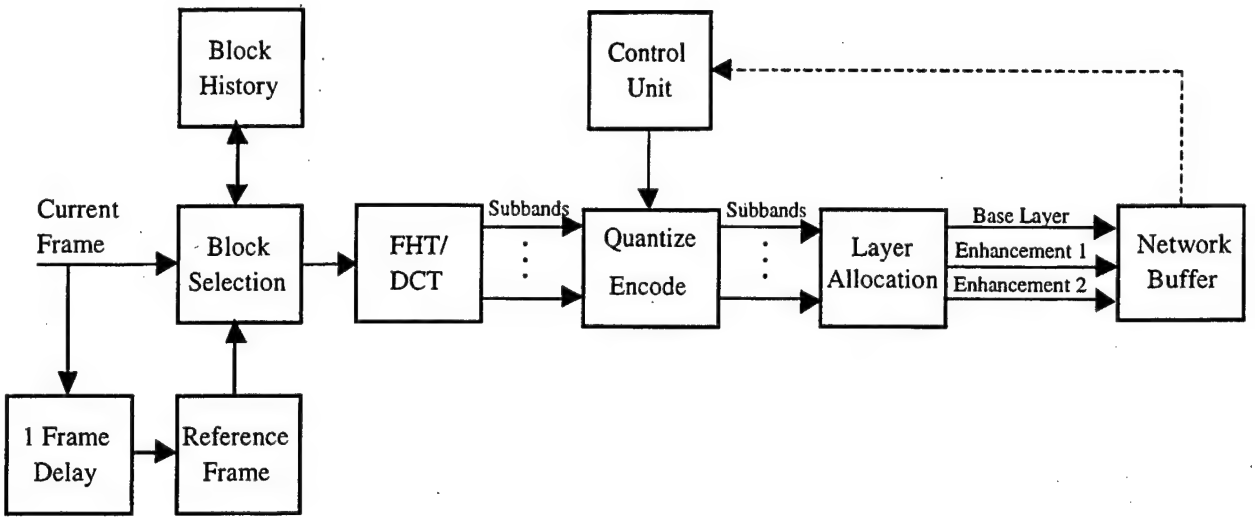

Figure IV.5: Functional Block Diagram of the Hybrid FHT/DCT Layered Coder.

1. Block Selection for Motion Compensation

Given the assumption of low activity video, temporal compression is provided through a simple block selection (updating) scheme that encodes only those macroblocks that show significant changes frame-to-frame. For low activity video, block selection yields only slightly inferior compression performance relative to motion prediction schemes [53]. Since interframe error propagation is greatly limited and intraframe error propagation is eliminated, it provides greater robustness. Block updating also voids the need for a locally decoded reference frame. This greatly simplifies the coder since an inverse quantization/transform loop is not required. Block selection is considered here solely with regard to video sequences. Static sequences exhibit little or no motion and consequently make little use of block selection. Indeed, most transmissions that occur during static sequences arise from the considerations presented in the next section that require the inclusion of a block-aging algorithm.

Motion is detected by applying a distance metric between successive frames. The distance between each macroblock in the current frame and its counterpart in the previous frame is calculated and the result compared to a threshold. To decrease computational

expense, the distance metric is applied to individual 8×8 blocks within the macroblock; the first block to satisfy the threshold triggers selection and ends the search, thus avoiding the expense of examining the remaining blocks. To further decrease computational expense, distance calculations are confined only to the luminous component of each pixel even if color components are present since the human visual acuity is more sensitive to changes in luminosity [6].

Since motion in VTC scenes tends to be confined to discrete objects within the scene, as opposed to scene motion caused by a camera pan, search efficiency is slightly affected by the order in which the individual blocks are examined. The more efficient approach is to maximize the distance between the first two blocks examined. As shown in Figure IV.6, two search patterns can be considered: a cross-pattern search that examines the upper left block followed by the lower right and a clockwise search starting from the upper left. In the test video sequences examined, for those macroblocks selected due to motion, the cross-pattern search resulted in a 2.5% decrease in the average number of blocks examined per frame compared to the clockwise search. The result was a net decrease of one block per frame. Of course, the decrease depends on motion content; with increasing motion, the difference becomes negligible.

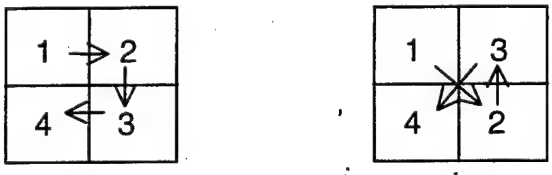

Figure IV.6: Block Search Order: a) Clockwise Search and b) Cross-pattern Search.

A much greater improvement is realized by using the cross-pattern search but changing the starting block of each macroblock each frame to match the anticipated motion at that point in the frame. Again, motion in VTC sequences is fairly confined. For example, a speaker shifts left to right and/or slightly up and down. Consequently, macroblocks tend to be selected in the same manner. Therefore, search speed is increased by having the coder store the identity of the specific block, termed the "anchor"

block, that caused a particular macroblock to be selected in the previous frames. For each macroblock in the new frame, the block selection algorithm starts from the anchor block. If the anchor block causes selection or if the macroblock is not selected, the anchor block identity is unchanged. If another block causes selection, the anchor block identity is updated. Using this search scheme produced an additional 20% improvement in the number of blocks searched and resulted in 10 fewer blocks searched per frame on average. A more complex approach not examined here is to remember the two blocks that most frequently caused selection and tailor the search accordingly. The resulting tailored search would be clockwise, counter-clockwise, or cross-pattern.

The distance metric employed is the non-normalized ASD given by [6]:

$$e_{ASD} = \left| \sum_{m=1}^{M} \sum_{n=1}^{N} \left(x_{m,n} - x_{m,n}^{R} \right) \right| \tag{IV.1}$$

where $x_{n,m}$ and $x_{m,n}^{R}$ represent the pixel intensities in the predicted and reference blocks, respectively. This expression of ASD differs from the form given by Eq. (III.10) in that the result is not normalized by the number of pixels and the reference macroblock is not offset. The non-normalized version is used since the normalization factor is easily included in the threshold value, saving the cost of a floating point division operation or, at least, a right-shift operation. The ASD is employed due to computational efficiency as it only requires additions and subtractions along with a single absolute value operation. SAD requires an equal number of arithmetic operations but requires $MN-1$ more absolute value operations. Further, since the ASD takes the absolute value of only the sum, it acts like an accumulator and provides a low-pass filtering effect that removes noise added to pixel intensities through video capture. Smoothing prevents spurious block selection in otherwise static screen regions that could occur in other metrics, such as SAD or MSE, where non-linear operations on a per-pixel basis tend to accumulate noise energy. This allows bandwidth to be more effectively devoted to regions of greater interest [45].

The relative selectivity of ASD and SAD was tested by determining the relative thresholds required to deliver approximately the same quality, as measured by pSNR, and

then comparing the resulting block selection rates and pattern. Examining Figure IV.7, a threshold index of below 8-10 was required to adequately capture motion scene motion. In this region, ASD selects 1-2 more macroblocks compared to SAD. However, examining the macroblocks selected confirmed that ASD tended to better capture speaker motion while SAD's selections were more diffuse. As a result, not withstanding the pSNR equivalence, video compressed using ASD was judged more visually pleasing. The difference in bandwidth appears negligible considering the vast decrease in computational effort required by ASD.

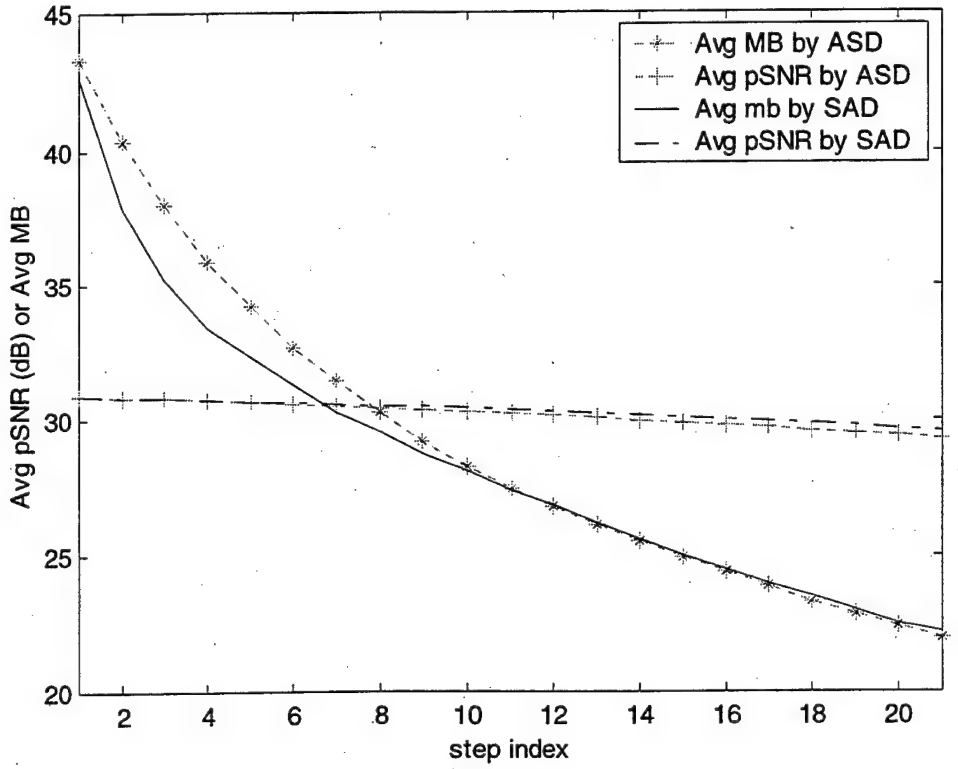

Figure IV.7: Comparison of ASD and SAD for Block Selection.

Two independent elements effect video quality and thus required bit rate: adequate motion detection to prevent "jerky" motion in the reconstructed video and controlling distortion introduced due to quantization. The goal in motion detection is to

select the *maximum* block selection threshold that adequately captures motion. In the video sequences examined, a threshold of 160 proved adequate. At this threshold value, an average of 24.8 macroblocks was selected per frame in test video sequences[11]. In practice, a user selectable threshold would prove beneficial by allowing the sender to compromise between motion selection and visual distortion given a set bit rate.

2. Aging Algorithm

Motion compensation using only block refreshment through the selection scheme described above presents some problems [45]. Consider an arbitrary macroblock whose content is changing due to motion within the frame. The macroblock travels from its initial state along some trajectory to a final state once the motion has stopped. At some point in the trajectory, the block selection algorithm forces an update to the macroblock. Once the final state is reached, hysteresis occurs if the distance between the final and updated states is not sufficient to force block selection; the distance differs by less than the threshold. In this case, the macroblock is not selected for updating, and the displayed macroblock at the receiver is left with a persistent error. Another problem occurs when new participants are allowed to join a VTC in progress (dynamic multicast) [45]. Since the coder is only transmitting those macroblocks selected due to motion, new participants receive a portion of the current scene. With low activity video, the end result is a patchy "disembodied" speaker. The final problem is the duration of error artifacts due to missing or corrupt packets at the receiver. Artifacts created in the active portion of the scene tend to last for only a single frame since block updates occur frequently. However, errors in less dynamic regions tend to persist longer since the frequency of updates is correspondingly lower. Due to lower motion content, each of these problems is of greater concern during static sequences since the block updating scheme selects either a few macroblocks to transmit, given an in-screen cursor, or none at all.

Coupling the block update scheme with an aging scheme that forces periodic updates of each macroblock alleviates these problems. The general principle is that the

[11] Actually more macroblocks are selected due to *forced* selections as covered in the next section.

coder tracks the time interval or age since each macroblock was last selected. If a macroblock's age exceeds a predetermined interval, that macroblock is selected by default. Aging thus guarantees a maximum period between macroblock updates. This bounds both the duration of hysteresis errors and visual artifacts caused by losses and errors during transmission. The bound also ensures that new viewers receive an entire frame in a timely manner.

Obviously, aging increases bandwidth requirements, but the impact is lessened by the manner in which macroblocks are selected through aging and the length of the aging interval. Spreading block selections evenly over time is desirable to avoid spikes in bit rate, which in turn requires a scheme that ages each block independently. Simply choosing to update a block after n frames pass without an update leads to an undesirable correlation in updates following each scene change. Even though motion within the scene tends to randomize updates to some extent, a sufficiently static background would still lead to correlation of a significant fraction of block updates. The worst case is represented by a scene change where the new scene is entirely static, such as a slide presentation. In this case, the bitrate spikes every n frames. Increasing the aging interval decreases bandwidth but increases the duration of visual errors and degrades response time for new participants.

The aging algorithm used in the coder does not track the age of each macroblock directly. Instead, each macroblock has an entry in an update table identifying the number of frames remaining until that macroblock must be updated. As each frame passes without an update, the entry is decremented by one. As each macroblock is processed for block selection in a given frame, the coder examines the macroblock's entry in the update table. If its corresponding entry has reached zero, the macroblock is selected for transmission. Otherwise, the distance metric is applied to determine if the macroblock should be selected due to motion. The order of the two events is important. Since the distance metric does not need to be calculated for those macroblocks selected due to aging, the result is a net decrease in the number of calculations required to select macroblocks for transmission. In either case, after a macroblock has been transmitted, a

new update is scheduled m frames in the future, where m is a discrete uniform random variable distributed in the range $[1,n]$. Pseudocode for this algorithm is listed in Figure IV.8. The update interval is initialized to 0 at the start of coding in recognition that all macroblocks in the first frame must be coded.

```
initialize update_table[99] to 0;

for each frame k

   % Process each macroblock in frame
   for each MB j = 1 to 99

      % Count down to next forced update
      update_table[j] -= 1

      % Check for forced update
      if update_table[j] = 0
         encode block
         update_table[j] = random update

      % Check for block selection
      else if distance(MB j) > threshold
         encode block
         update_table[j] = random update
      end

   end

end
```

Figure IV.8: Pseudocode for Aging Algorithm.

Using a uniform distribution to schedule updates smoothes block selections over n frames and decorrelates the selection of individual macroblocks through aging. Choosing aging intervals randomly also prevents events, such as scene changes, from correlating updates and generating spikes in bit rate. The value chosen for n controls the tradeoff between additional bandwidth required and coder responsiveness. For a given value of n, the number of additional macroblocks selected through aging per frame N_a is

$$N_a = \frac{2N_{MB}}{(n+1)} \qquad \text{(IV.2)}$$

where N_{MB} is the number of macroblocks in the frame. Actually the bandwidth impact is lower since some of the blocks selected via aging would have been selected anyway due to scene motion.

For the video sequences examined in this work, n was set to 20. This value offers an acceptable compromise between bandwidth, corresponding to an additional 9.43 macroblocks per frame, and responsiveness. New VTC participants are guaranteed to receive a complete frame after 2 seconds, at 10 fps, and visual errors are bounded by the same value.

3. Layering Strategy

Macroblocks selected for transmission are decomposed into layers using a wavelet transform. Since the selection process takes place before the transform stage, the transform is only applied to those macroblocks requiring transmission. A wavelet-based approach was chosen since frequency decomposition offers the most flexibility in populating layers. A macroblock may reasonably be decomposed into as many as sixteen 2×2 subbands, using a uniform decomposition, which then may be combined in various manners to create an arbitrary number of layers (up to sixteen). The challenge is in determing an appropriate number of layers and apportionment of the frequency content within the macroblock across those layers.

As layers are hierarchical in importance, layer assignments should map frequency content to that hierarchy in a manner consistent with perceptual importance. Just as important, the bit rate allocation resulting from the layer assignments should be segragated such that dropping a layer offers the potential for decreasing congestion. In practice, meeting these expectations with a single layering scheme proved impractical. Therefore, two specific layering schemes were required: one for video sequences and one for static presentation slide sequences.

For both types of sequences, layering is accomplished through application of the fast Haar transform (FHT) to each selected macroblock. The FHT is the simplest possible wavelet algorithm [60] and is described by

$$x_1^a(n) = \frac{1}{2}\left(x_0^a(2n) + x_0^a(2n+1)\right) \quad \text{(IV.3)}$$

$$x_1^d(n) = \frac{1}{2}\left(x_0^a(2n) - x_0^a(2n+1)\right) \quad \text{(IV.4)}$$

where x_0^a is the original data vector, and vectors x_1^a and x_1^d are the average and detail decomposition vectors, respectively. The FHT has several desirable properties with regard to minimizing coder complexity. First, the FHT is a real transform, so no complex arithmetic is required and storage is simplified. Second, the FHT is not computationally demanding as its application requires only addition, substraction, and left- and right-shifts. Finally, unlike more sophisticated wavelet transforms, the FHT does not require extending or padding the data set. However, the simplicity of the FHT can lead to blocking artifacts at high compression levels since the average and detail calculations are confined only to contiguous pixels.

Since video information is two-dimensional, Eq. (IV.3) and Eq. (IV.4) can be applied to each dimension idependently, resulting in four uniform subbands as discussed in Section III.F. A key difference from that discussion is that the average and detail equations are applied to individual macroblocks instead of the entire frame. The resulting average (LL) subband and the three detail subbands (HL, LH, and HH) are each 8×8 in size. The actual operations required to generate each subband and the physical signficance of each subband are given in Table IV.2.

Subband	Detail	Horizontal Operation	Vertical Operation
LL	Lowpass	Average	Average
LH	Horizontal	Average	Detail
HL	Vertical	Detail	Average
HH	Diagonal	Detail	Detail

Table IV.2: Significance and Determination of Wavelet Subbands.

The coder restricts the number of layers to three. The decision to consider no more than three layers was driven by the limited bandwidth available. Each layer

consumes an equal amount of bandwidth in overhead. While a greater number of layers offers more flexibility in managing quality and congestion, at 64-96 kbps, three layers appears to be the limit in terms of producing layers that provide a perceptible improvement in quality.

The initial layering strategy considered for both the video and the static slide sequences performs only a first order analysis of each selected macroblock, generating the subbands listed in Table IV.2. Each subband generated is assigned to a layer as shown in Table IV.3. The layer assignments are intended to promote a graceful increase in quality by progressively adding frequency content. The base layer is essentially a lowpass-filtered version of the original macroblock, and the two enhancement layers successively add in higher frequency details. Since the LL subband retains many of the perceptual properties of the original macroblock, the LL subband is transformed further using the 2-D DCT. The additional transform allows the LL subband to be processed using JPEG, an approach that exploits that standard's emphasis on maximizing retention of the most perceptually relevant information.

Layer	Subband(s) Included
Base	LL
1st Enhancement	LH, HL
2nd Enhancement	HH

Table IV.3: Preliminary Layer Assignments

Preliminary results for the initial layering approach were disappointing. With regard to video sequences, the base layer gives acceptable quality and the first enhancement layer produced a marked improvement in quality. However, the bit rate allocated to the second enhancement layer by this assignment scheme was small (< 10%), and application of the layer only occasionally produced a perceptible improvement in quality. For static slide sequences, the situation is reversed. Slides consist of text and line drawings, which exhibit a different frequency characteristic than motion video. The preponderance of sharp edges, in all directions, increases the relative importance of

higher frequency content in these frames relative to motion video frames. As a result, the hierarchy given in Table IV.3 is reasonable for motion video but unsuited for static sequences. Due to the absence of high frequency content, text and block diagrams were blurry and indistinct. Even adding the first enhancement layer only yielded a marginal improvement. Indeed, only the final addition of diagonal detail produced acceptable quality.

The results indicate that a frequency-based hierarchical scheme designed for motion video is unsuitable for static sequences. Although examined further below, the converse also appears to be true. Therefore, separate layering schemes were formulated for each sequence. The coder deduces the type of sequence present and applies the appropriate layering scheme.

The ad hoc approach presented above indicates the need for a more general technique for determining an appropriate layering structure for a video stream. The problem is to determine, given that n layers are desired, to what degree a selected macroblock is decomposed and how the resulting subbands are allocated to each layer. Here, we propose a variant of the split-and-merge algorithm [73] applied at the macroblock level. Instead of applying the algorithm in the spatial domain to identify regions of equivalent activity, the algorithm is applied to selected macroblocks in the frequency domain to identify regions of similar energy and perceptual content. Essentially, the macroblock is split into equal segments using the FHT, subbands of approximately equal variance are grouped, and the resulting regions are allocated to individual layers. At this point, dynamically changing the layering structure is not permitted.

Given a representative video sequence, the first step of the algorithm is to split each macroblock using the FHT. The macroblock is split into equal subbands by recursively applying the FHT to each subband until the desired number of subbands is created. For example, a first order decomposition of the macroblock creates four 8×8 subbands (LL, LH, HL, HH). A second order decomposition of each of these subbands creates sixteen 4×4 subbands as shown in Figure IV.9. Continuing the example, a second

order decomposition of the LL subband results in the LLLL, LLLH, LLHL, and LLHH subbands. Likewise, a third order decomposition produces 64 2×2 subdands. In practical application, stopping at a second order decomposition proved sufficient for three layers.

Using the representative video sequences, the variance of the coefficient set comprising each subband is determined across all frames of video. Using subband variance as a metric to form layers offers two benefits. First, with motion video, variance appears to have an inverse relationship to spatial frequency and thus perceptual importance. Therefore, differences in variance provide a convenient mechanism for assigning subbands to a layered hierarchy. Second, grouping subbands with a similar variance is convenient since each group can employ a common quantizer. Several quantizer schemes allocate bits by varying quantizer step size in inverse proportion to variance. This approach uses variance as indication of the dynamic range exhibited by the coefficients. One such scheme, described later, apportions bits in an attempt to balance distortion introduced across each subband [48].

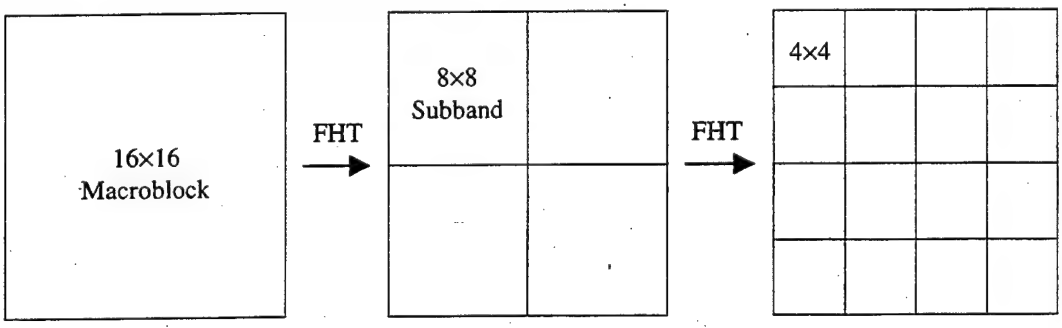

Figure IV.9: Splitting a Macroblock into Uniform Subbands.

The subband variances, computed using several test video sequences, after first order decomposition, are shown in Table IV.4. The subband variances after a second order decomposition are shown in Table IV.5. Subband variance provides a good indication of energy concentration within each subband. Since the video images are lowpass, the energy is concentrated in the lowest subband as shown in Table IV.4. By extension, the subband variance also provides an indication of relative perceptual importance, an observation that allows subband variance to dictate layer assignments. A

second order decomposition further differentiates the frequency content found in the first order subbands. For example, after a second level decomposition of the LH subband, energy is now concentrated in the LHLL and LHLH subbands. Values in Table IV.5 resemble the transpose of Figure III.4 and demonstrate that subband variance strongly tracks the visual components in the macroblock. This strengthens the argument for using subband variances to make layer assignments in a hierarchical manner for video sequences.

σ^2_{LL}	σ^2_{LH}		2891	52.0
σ^2_{HL}	σ^2_{HH}	=	73.3	12.4

Table IV.4: Subband Variances after a First Order Decomposition (Motion Video).

σ^2_{LLLL}	σ^2_{LLLH}	σ^2_{LHLL}	σ^2_{LHLH}		2702.0	57.7	19.2	21.6
σ^2_{LLHL}	σ^2_{LLHH}	σ^2_{LHHL}	σ^2_{LHHH}		117.4	12.5	4.5	6.8
σ^2_{HLLL}	σ^2_{HLLH}	σ^2_{HHLL}	σ^2_{HHLH}	=	27.7	6.0	2.2	2.8
σ^2_{HLHL}	σ^2_{HLHH}	σ^2_{HHHL}	σ^2_{HHHH}		31.5	8.0	3.2	4.2

Table IV.5: Subband Variances after a Second Order Decomposition (Motion Video).

After variance data has been gathered for each subband at the desired level of analysis, the next step is to group adjacent subbands exhibiting similar variances. The criterion suggested by [73] is to group adjacent subbands k_1 and k_2 with variances $\sigma^2_{k_1}$ and $\sigma^2_{k_2}$, respectively, when

$$\left|\log\left(\frac{\sigma^2_{k_1}}{\sigma^2_{k_2}}\right)\right| \leq \Delta\sigma^2 \qquad \text{(IV.5)}$$

where

$$\Delta\sigma^2 = \frac{1}{N_b}\log\left(\frac{\sigma^2_{\max}}{\sigma^2_{\min}}\right), \qquad \text{(IV.6)}$$

and σ^2_{max} and σ^2_{min} represent the maximum and minimum variances found among the subbands. The parameter N_b represents the total number of subbands.

Grouping of subbands based on the variances in Table IV.5 results in the partitions shown in Figure IV.10. Assuming that each subband is independent, the variance of each partition P_k is simply the sum of the variances for the subbands k_i comprising that subband:

$$\sigma^2_{P_k} = \sum_{k_i \in P_k} \sigma^2_{k_i} . \qquad \text{(IV.7)}$$

Since the subbands comprising each partition have similar variances, each partition can be quantized using the same scheme such that quantization errors are spread uniformly among the subordinate subbands.

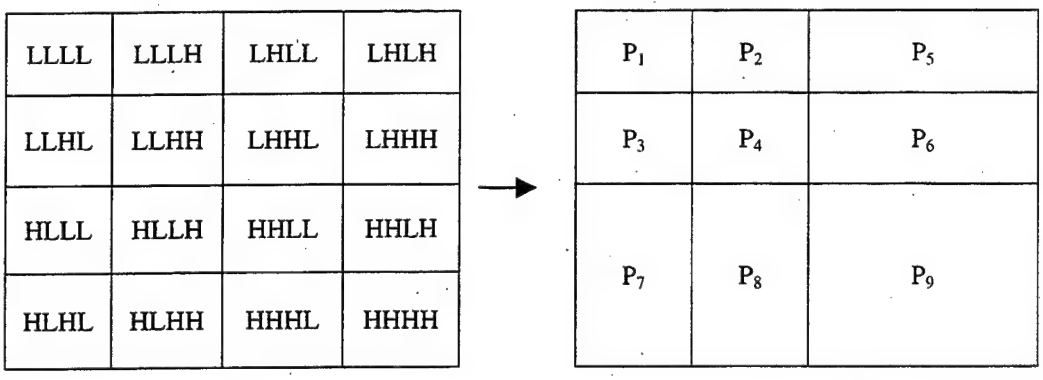

Figure IV.10: Partitions Resulting from Merge Algorithm.

Next, the resulting partitions are assigned to layers L_j until the requisite number of layers are created using the following set of heuristic rules:

Rule 1: No layer may have a greater variance than any lower layer. That is, given N layers,

$$\sigma^2_{L_0} > \sigma^2_{L_1} > \cdots > \sigma^2_{L_{N-1}} . \qquad \text{(IV.8)}$$

Rule 2: Layers must be populated in order of increasing frequency content. A layer may not contain a partition of lower frequency content than any layer below it.

Rule 3: Partitions that meet the criterion given by Eq. (IV.7) are assigned to the same layer even if the partitions are non-contiguous.

Rule 4: Partitions are applied to layers in a symmetric fashion.

Rule 5: If more than two subbands comprising a coarser subband remain as partitions after applying the above rules, group all of the partitions comprising the coarser subband together into one partition.

Rule 6: If one or more partitions are moved between layers, as required to achieve a more balanced distribution of bit rates or quality, move the partition(s) with the lowest variance if promoting to a higher layer and the partition(s) with highest variance if demoting to a lower layer.

The reasoning behind these rules stem from the requirements stated for layered coder design at the start of this section. Rule 1 ensures that no upper layer receives a greater bit allocation than the lower layers. This provides a more logical sequence to the layer hierarchy since the lower layers will make a greater contribution to reconstructed quality, and quality loss due to layer dropping is more gradual. Rule 2 matches the layer hierarchy to the observed frequency dependence displayed by the human visual system (HVS) and ensures a more graceful degradation in quality during periods of congestion. Rule 3 simplfies quantizer design by allowing non-contiguous partitions to use the same quantization scheme. Rule 4 ensures that neither horizontal or vertical detail dominate a partially reconstructed frame. A lack of balance between these components distorts the image and causes scene elements to appear elongated. Simplifying coder design and

minimizing processing delay are the main considerations for Rule 5. Each distinct partition or subband transmitted requires overhead within the bit stream for the decoder to correctly position the contributions. A greater number of subbands also complicates quantizer design and rate control. Concatenating the single subband partitions into their coarser, parent subband offsets these concerns and reduces the computational burden required to transform the macroblock since an analysis step is dropped.

Rules 1-5 help determine an effective layering scheme for motion video. However, implementation provides the final test of the effectiveness. Two problems may result during implementation as discovered in the first ad hoc approach attempted. The resulting bit rate for a layer may be small such that bitstream overhead is too high. Or a layer may appear to offer a negligible impact of reconstructed quality. In either case, the solution is to reduce the number of layers by concatenating the ineffectual layer with an adjacent layer or to move partitions between layers. The latter situation is covered under Rule 6, which provides guidance for moving partitions between layers without violating the other rules.

Application of these rules to the partitions shown in Figure IV.10 resulted in the final layering scheme for motion video sequences shown in Figure IV.11. The LL subband is assigned to layer I and further transformed via DCT as previously discussed. The HH subband is assigned to layer III in its entirety. The HL and LH subbands are further decomposed. The resulting subbands are partitioned and assigned to layers II and III. The layer assignments in Figure IV.11 also provide the basis for the quanization scheme discussed in the next section.

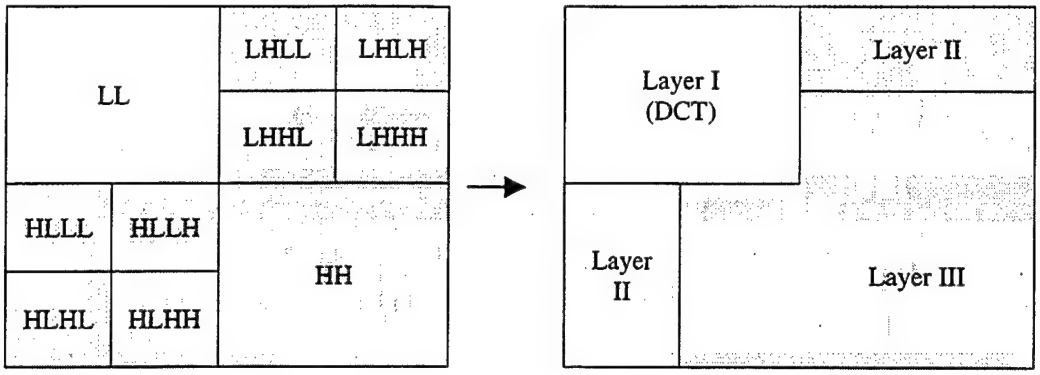

Figure IV.11: Final Layering Scheme for Motion Video Sequences.

The generalized layering scheme presented above is biased for motion video sequences. Consequently, the layering scheme presented in Figure IV.11 is not suitable for static slide sequences. Static slide sequences show a much greater dependence on higher frequency components for perceptual recognition since text and line drawings have a much higher preponderance of edge detail. Any hierarchical scheme based on the lowpass nature exhibited by images yields a blurred reproduction with only the lower layers and gives satisfactory results only when the high frequency layers are added. For example, applying the motion video layering scheme to slides containing text and line drawings only gives acceptable results when all three layers are received. Obviously, this defeats the purpose of layering video. Therefore, a different layering scheme is appropriate if the video stream is to include both types of sequences.

Although the general layering scheme presented above is not applicable to static slide sequences, application of the split-and-merge algorithm is still meaningful. The variances exhibited by the subbands generated after a first and second level analysis of slide sequences consisting of text and line drawings is shown in Table IV.6 and Table IV.7, respectively. Comparing these values to those for the motion video sequences given earlier, it is evident that energy is much more evenly distributed among the different subbands. The result promotes a much more complex relationship between variance and perceptual importance which is demonstrated in the close interdependence between the various subbands.

σ^2_{LL}	σ^2_{LH}	=	2613	1216
σ^2_{HL}	σ^2_{HH}		1209	610

Table IV.6: Subband Variances after a First Level Decomposition (Slide Sequence).

σ^2_{LLLL}	σ^2_{LLLH}	σ^2_{LHLL}	σ^2_{LHLH}		1392	600.5	404.4	441.3
σ^2_{LLHL}	σ^2_{LLHH}	σ^2_{LHHL}	σ^2_{LHHH}		456.7	159.2	184.3	185.7
σ^2_{HLLL}	σ^2_{HLLH}	σ^2_{HHLL}	σ^2_{HHLH}	=	536.2	128.3	162.2	148.1
σ^2_{HLHL}	σ^2_{HLHH}	σ^2_{HHHL}	σ^2_{HHHH}		423.3	121.1	141.7	157.8

Table IV.7: Subband Variances after a Second Level Decomposition (Slide Sequence).

Applying the split-and-merge algorithm results in the partitions shown in Figure IV.12. Using the layer assignment rules outlined above, partitions P_1, P_2, and P_4 are assigned to the base layer. However, reconstruction based solely on the base layer gives very poor results. Even adding partitions P_3, P_5, and P_6 fails to achieve acceptable results even though such an arrangement includes a large portion of the energy contained in the macroblock. Therefore, unlike in the motion video case, variance alone provides a very poor guide to determining perceptual relevance. Instead, achieving acceptable reconstruction starting with the base layer requires contributions from each of the 8×8 subbands. In practice, the layering scheme shown in Figure IV.13 was found to be suitable. The base layer consists of those 4×4 subbands containing the most significant details as determined by variance. Although in motion sequences the LLLL subband is expected to have a lowpass frequency characteristic consistent with the original macroblock, this does not hold true with the static sequences. Therefore, application of the DCT provides no additional benefit. The remaining subbands are divided between the remaining layers in order of increasing frequency content.

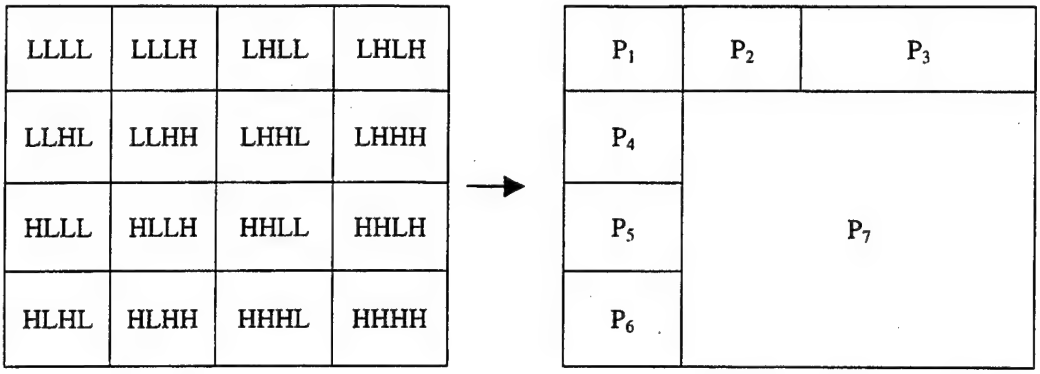

Figure IV.12: Partitions Resulting from Merge Algorithm (Slide Sequence).

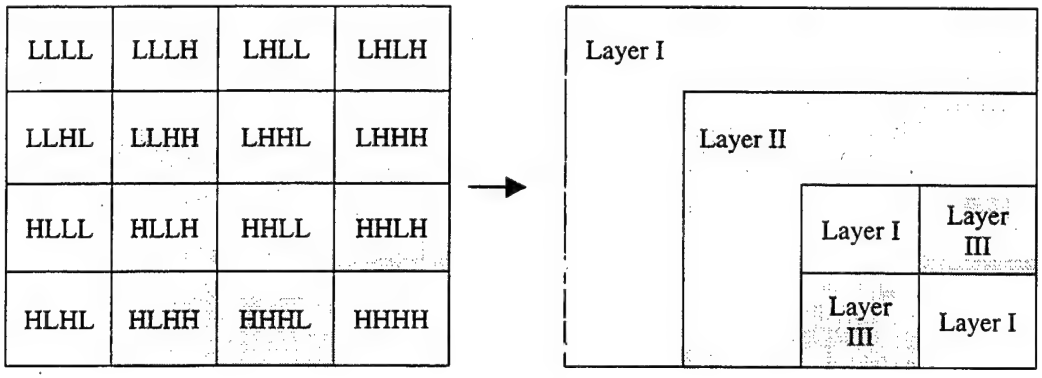

Figure IV.13: Final Layering Scheme for Static Slide Sequences.

Although the partitions in Figure IV.12 do not directly lead to a satisfactory layering arrangement, continuing the examination does lead to a simple quantization scheme. After merging partitions with similar variances, the partitions have been reduced to those shown in Figure IV.14. Although partitions P2 and P3 are not close enough for merging, given Eq. (IV.5), they are sufficiently close in variance such that the simplicity gained by quantizing both bands together balances any possible sub-optimal bit allocation. The final partitions, for the purpose of quantization, are shown in Figure IV.15.

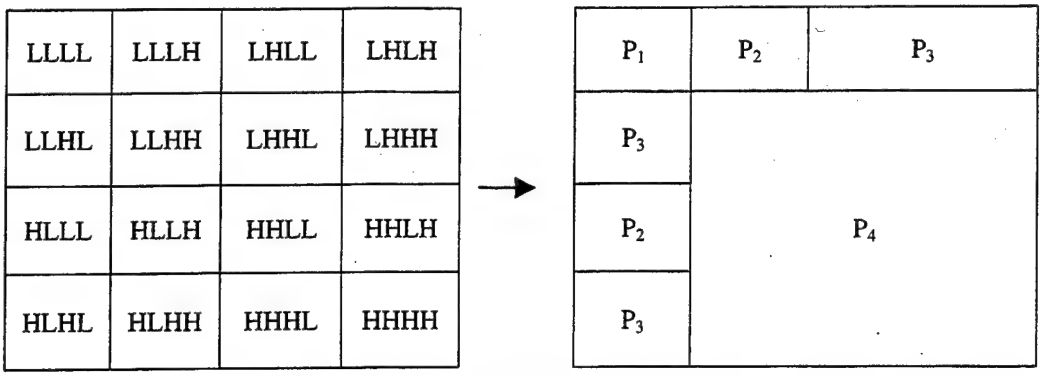

Figure IV.14: Partitions Remaining After Merging Similar Non-Contiguous Partitions.

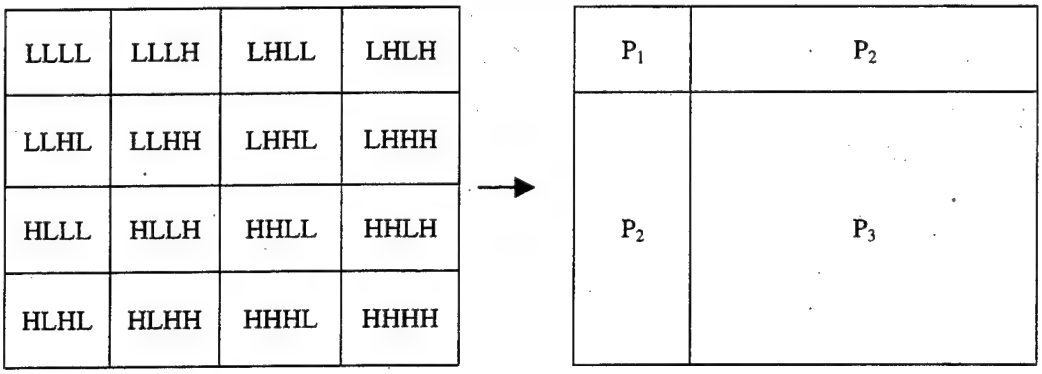

Figure IV.15: Partitions for the Purpose of Quantization.

Since two different layering schemes are used, the coder requires some criteria for determining the type of video is present. The determination is made following each scene change. The coder judges that a scene change has occurred if the number of macroblocks selected exceeds some threshold. After examining the block selection statistics for motion video, selecting a threshold three standard deviations above the mean block selection rate was high enough to avoid spurious scene change detections. If a scene change has occurred, the coder examines the number of macroblocks selected due to motion in the next frame. If the value is zero, the current sequence is assumed to be static since obviously no motion has occurred within the scene. Otherwise, the sequence is assumed to be a motion video sequence.

4. Quantization and Lossless Coding

After the transform stage, individual subbands are quantized and losslessly coded according to their layer assignment (motion video sequences) or partition assignment (static sequences). The main difference is that the base layer for motion video sequences is encoded using the JPEG standard. Otherwise, uniform quantization is used with a single step size for each layer/partition followed by Huffman coding.

The quantization and coding stage for motion video macroblocks is shown in Figure IV.16. The LL subband coefficients are quantized and encoded using the luminance quantization array and luminance VLC table suggested in [75]. This process is summarized in Chapter III.

The remaining subbands are uniformly quantized using a fixed quantizer step size for all coefficients in that subband. The value of the quantizer step size is set independently for Q1 and Q2, and all subbands entering a particular quantizer use a common step size.

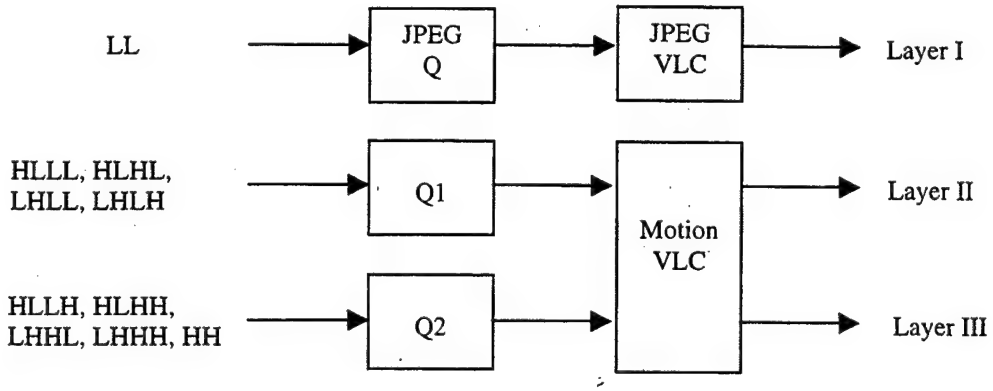

Figure IV.16: Quantization and Coding for Motion Video Macroblocks.

Unlike in JPEG encoding, zig-zag scanning of the quantized FHT coefficients provides no apparent coding gain. Instead, trials indicated that a simpler horizontal raster scan was adequate for all bands except the HL subband. The HL subband showed a slight preference for a vertical raster scan, which seems consistent given the frequency orientation of this band. The scan orders are summarized in Table IV.8, where the scan

order applies to the subband indicated as well as all child subbands. The LL entry pertains only to coding of static macroblocks and is included for completeness.

Parent Subband	Scan Order
LL	Raster
LH	Raster
HL	Vertical Raster
HH	Raster

Table IV.8: Scan Order for Encoding Quantized Coefficients.

After scanning, each non-zero coefficient is losslessly coded using a Huffman VLC code. The coding scheme chosen mirrors the 3-D event structure employed by the H.263 coding standard. Each non-zero coefficient is replaced by an equivalent event described by three parameters [56]: {LAST, RUN, LEVEL} where LAST indicates whether there are any more non-zero coefficients in the current subband; RUN indicates the number of successive zeros that precede the non-zero coefficient; and LEVEL represents the non-zero magnitude of the quantized coefficient. Each event maps to a VLC codeword to which a sign bit is appended to represent the sign of the coefficient. A VLC table was derived for motion sequences using a series of representative test sequences [74].

The quantization and coding stage for static macroblocks is shown in Figure IV.17. The major difference compared to motion macroblocks is that JPEG is not employed. Instead, the sixteen subbands are supplied to one of three independent uniform quantizers, Q1, Q2, and Q3, each with a fixed quantizer step size. After quantization, each non-zero coefficient is replaced by a 3-D VLC codeword as described above although a different VLC table is employed. Again, the VLC table was developed from a series of representative sequences [74].

Neither Figure IV.16 nor Figure IV.17 indicates the presence of the control signal from the Control Unit shown in Figure IV.5. The control signal allows manipulation of

the quantizer step sizes, or a scaling factor in the case of the JPEG quantizer, as required by a rate control scheme. Rate control schemes are covered later in this chapter.

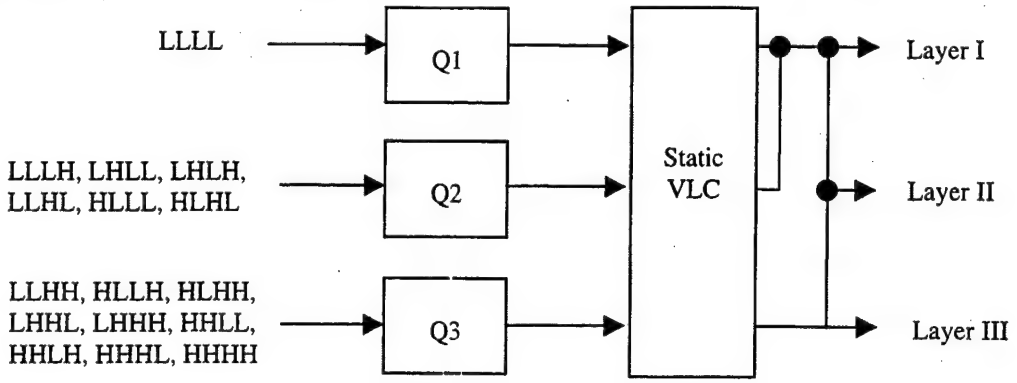

Figure IV.17: Quantization and Coding for Static Macroblocks.

D. RESULTS

This section includes some example video traces for a short video segment consisting of 100 frames of a single speaker followed by 50 frames of a presentation slide filled with line diagrams and text. A sample frame from each sequence is shown in Figure IV.18 and Figure IV.19. Each shows the original frame and the reconstructed frame with only the base layer received, the base layer and the first enhancement layer received, and all layers received. With the exception of scene changes, the coder employed no rate control for these sequences; a single set of quantizers is used for each sequence and not varied during the run. During a scene change, the first new frame of the scene is heavily compressed to avoid spikes in the outgoing bit rate. The video quantizers employed produced an average bit rate of 80 kbps for the video sequence and 40 kbps for the static sequence although the bit rate would be expected to vary locally.

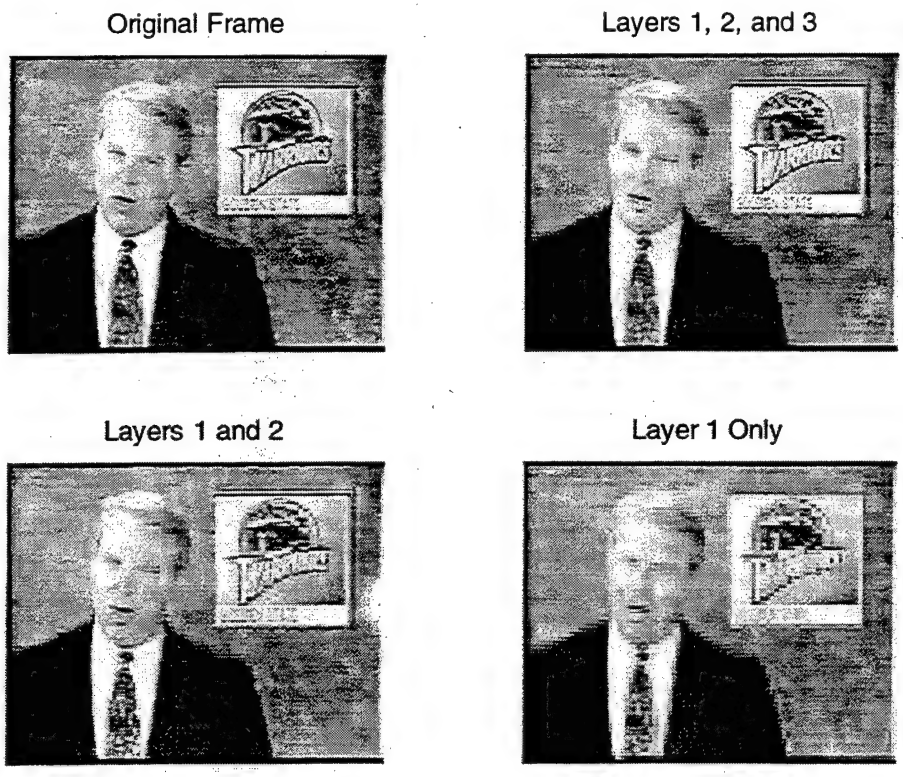

Figure IV.18: Original and Reconstructed Frames From a Motion Video Sequence.

Figure IV.20 and Figure IV.21 show the bit rate trace for the combined sequences and the plot of pSNR as a measure of reconstructed video quality (see Eq. (III.7)). The granularity in bit rate offered by a layered video hierarchy is evident in Figure IV.20; as congestion occurs, the lower layers could be retained while preserving most of the quality. The bit rate ratio among layers is approximately 5:3:2 for both sequences. As expected, the bit rate for the static sequence is much lower since the bit rate results solely from macroblock aging. For this reason, rate control is not of significant benefit for the static sequences. Using a pointer within the overhead slide would result in macroblocks selected due to motion and increase the bit rate slightly, but bit rate would still not reach the level displayed for the motion video sequence.

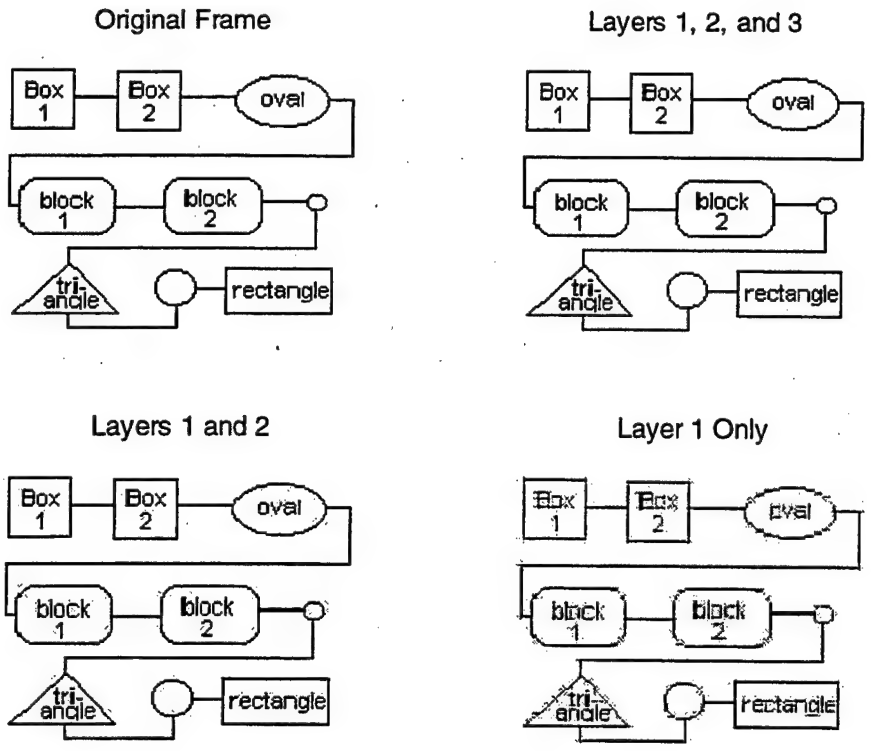

Figure IV.19: Original and Reconstructed Frames From a Static Video Sequence.

Figure IV.21 illustrates the progressive improvement in quality as additional layers are added to the base layer. At the beginning of each sequence, quality in terms of pSNR improves sharply over the aging interval following a scene change. After this period, quality is observed to remain relatively flat for each sequence regardless of the number of layers as expected since no attempt is made to vary bit rate. For the motion video sequence, the base layer provides a smoothed but acceptable display. Text in the frame is not readable, but the speaker's movements are easy to follow. Adding the first enhancement layer improves sharpness and adds a 4 dB improvement in pSNR although small text is still difficult to discern. The second enhancement layer only adds 1-2 dB improvement but small text is finally readable and other features with fine edges are sharper. With static video, the role of the enhancement layers is even more dramatic.

Even though most of the macroblock's energy is included in the base layer and contributions from each frequency band are included, the base layer still shows a large degree of smoothness although the shapes are readily identifiable. Adding the first enhancement layer adds a 7 dB improvement and dramatically improves sharpness. The final layer, even though the bit rate contribution is the smallest of the three layers, almost doubles the pSNR, and the reconstructed frame is virtually identical to the original frame.

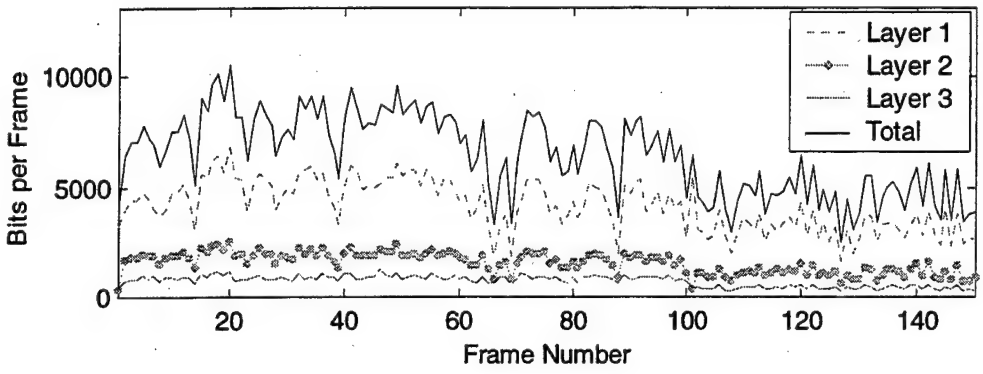

Figure IV.20: Bitrate per Frame for the Layered Video Sequence.

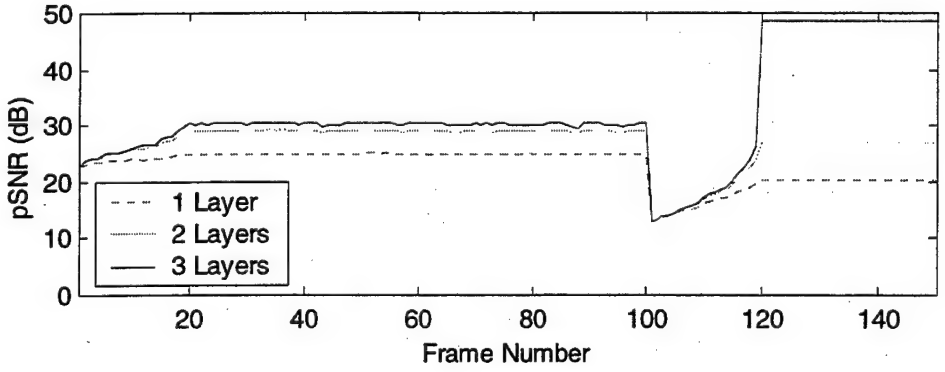

Figure IV.21: Reconstructed pSNR for the Layered Video Sequence.

E. SIMPLE LAYERED-VIDEO RATE CONTROL

Compressed video is variable bit rate by nature since compression gain varies based on scene activity and complexity. However, transmission channels inevitably require some constraints on bit rate because of channel capacity or QoS constraints. Most commonly, bit rate is constrained to maintain a constant rate or to maintain a constant local-average bit rate over time. Many factors affect bit rate, but the most important is the tradeoff between quantizer step size and image fidelity. A larger step size results in a lower bit rate and a larger amount of distortion. Reducing the step size increases the bit rate but reduces the amount of distortion. Rate control, therefore, requires evaluation of the rate-distortion relationship created by a particular coder design. The rate control problem may be posed in terms of the rate-distortion relationship. The goal of the encoder is to minimize distortion D subject to a bit constraint R_c, i.e., $R < R_c$ [53]. This problem is solved using Lagrangian optimization by expressing a cost function in terms of a distortion term weighted against a rate term [48]. The optimal solution is one that minimizes the cost function J, given by

$$J = D + \lambda R, \tag{IV.9}$$

where λ is the Lagrange multiplier. Expressing distortion as a function of rate, $D(R)$, and differentiating on both sides with respect to R to find a minimum results in

$$\frac{\partial J}{\partial R} = \frac{\partial D(R)}{\partial R} + \lambda = 0, \tag{IV.10}$$

which indicates that each Lagrange multiplier λ yields a particular optimal solution. Each tangential point on the rate-distortion curve therefore corresponds to an optimal solution for a particular rate constraint. Figure IV.22 shows a possible rate-distortion curve and an optimal solution for a bit rate of R_0. While the true rate-distortion curve is guaranteed to be convex [48], the operational curve is influenced by the coder design, including the motion-prediction scheme employed, the quantizer design, and lossless coding gains. Therefore, rate control schemes tend to only approximate the rate-distortion relationship when determining a method for varying quantizer step size to achieve the desired bit rate.

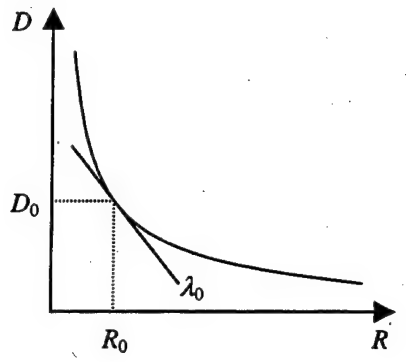

Figure IV.22: Rate-Distortion Curve and a Possible Optimal Solution.

With any rate-control scheme, two issues are of importance. First, changes to quantizer step size must be communicated to the decoder, which adds to the coder's overhead depending on how often the parameter is changed. Second, rate-control schemes must be kept reasonably simple for real-time applications to minimize coding delay.

Numerous feedback control schemes for rate control have been proposed that track actual bit allocation in some manner and use feedback to vary quantizer step size. The H.261 standard [67] suggests an approach described as liquid level control [6]. The H.261 reference coder examines the output buffer every 11 macroblocks. If the buffer is full, quantizer step size is increased. If the buffer is nearing empty, quantizer step size is decreased. H.261 leaves the actual rate control scheme up to the designer. One feasible approach is the feedback control scheme proposed by Choi and Park [76] that controls the Lagrange multiplier λ based on the output buffer state. Low-delay rate control approaches have been described by Telnor Research [55] and Ribas-Corbera and Lei [77] for H.263 and H.263+, respectively. The Telnor approach linearizes the relationship between quantizer size and bit rate. At the start of each frame, the coder determines the deviation between the bits allocated to the last frame $B_{i-1}$and the target bit allocation $\overline{B}$, i.e.,

$$\Delta B_1 = B_{i-1} - \overline{B} . \qquad \text{(IV.11)}$$

The coder also attempts to allocate an equal number of bits to each macroblock while encoding the current frame and tracks this deviation using the relationship

$$\Delta B_2 = B_{i,mb} - \frac{n_{mb}}{N_{MB}} \overline{B}, \tag{IV.12}$$

where n_{mb} represents the sequence number of the current macroblock and N_{MB} the total number of macroblocks. Then, at the beginning of each new macroblock, the coder updates quantizer step size based on these deviations:

$$Q_{mb} = \overline{Q}_{i-1}\left(1 + \frac{\Delta B_1}{2\overline{B}} + \frac{12\Delta B_2}{R}\right), \tag{IV.13}$$

where R is the allocated channel bit rate and $\overline{Q}_{i-1}$ is the average quantizer size in the previous frame. Telenor's approach gives an equal weighting to each macroblock. The approach taken by [77] is similar but computes an optimal quantizer step size for each macroblock within the bit budget using the variance exhibited by each macroblock as well as heuristic weight indicating the perceptibility of decode artifacts.

The issue of rate-control for layered video has not been well addressed in the literature. The rate-control problem is somewhat complicated by the multi-dimensional aspect of the rate-distortion curve expressing overall distortion as a function of an n-dimensional set of quantizers. In the coder presented here, the bit rate depends on a set of three quantizers. Two approaches are presented below. The first is based on a traditional rate-distortion approach that assumes that both rate and distortion for each layer are additive. The second approach uses vector quantization to reduce the dimensionality of the control problem and approximates an optimal rate-distortion curve.

1. A Rate-Distortion Approach

For a layered coder, separate quantizers are employed for each layer. Assuming that distortion for each layer i is additive, the rate control problem becomes minimizing

$$D = \sum_{i=0}^{N-1} D_i \tag{IV.14}$$

subject to the constraint

$$\sum_{i=0}^{N-1} R_i \le R. \tag{IV.15}$$

The assumption that each layer behaves independently allows the cost function to be rewritten as [48]

$$J_i(R_i) = D_i(R_i) + \lambda R_i, \tag{IV.16}$$

$$J = \sum_{i=0}^{N-1} J_i. \tag{IV.17}$$

Since the costs are additive, J is minimized when each J_i is minimized. Taking the derivative of Eq. (IV.16) to find the minimum results in

$$\frac{\partial J_i}{\partial R_i} = \frac{\partial D_i(R_i)}{\partial R_i} + \lambda = 0. \tag{IV.18}$$

Therefore, a particular bit rate R is optimal when each R_i corresponds to points with the same slope on their respective rate-distortion curves.

The distortion D_i introduced by quantization is related to the rate R_i by [78]

$$D_i(R_i) = C_i \sigma_i^2 2^{-2R_i}, \tag{IV.19}$$

where C_i depends on the pdf of the quantized variable, and σ_i^2 is the variance of the input values. Using this relationship, the Lagrangian method yields the following optimal solution [48],

$$R_i = \overline{R} + \log_2 \frac{\sigma_i}{\rho}, \tag{IV.20}$$

where $\overline{R} = R/N$ is the mean bit rate per layer, N is the number of layers, and

$$\rho = \left(\prod_{i=0}^{N-1} \sigma_i \right)^{1/N}. \tag{IV.21}$$

The allocation given by Eq. (IV.20) ensures that each quantizer has the same average distortion.

Using Eq. (IV.20), one possible frame-based rate control scheme could be implemented as follows. First, establish the bit allocation R for the current frame. Then, calculate the bit allocation for each layer using Eq. (IV.20). Finally, allocate the bits

evenly per coefficient for each layer. The bits allocated per pixel are used to calculate the quantizer step size. The following relationship is suggested to calculate quantizer step size for low bit rate video traffic [79]:

$$Q_i = \sqrt{\frac{e}{\log_e 2}\frac{\sigma_i^2}{R_i}}, \tag{IV.22}$$

where e is Napier's constant and σ_i^2 is the variance of subband i. For macroblocks that use multiple quantizer step sizes, such as in JPEG coding, the result is used to establish the average step size for the macroblock.

There are several drawbacks with this approach for rate control. The most important is that ensuring average distortion at each quantizer does not account for the perceptibility of errors in different frequency bands, and allocating errors in a different manner could provide more *optimal* results perceptually. The allocation also depends on knowledge of the variances exhibited by each layer. Although representative variances may be calculated *a priori* using test sequences, more accurate allocation requires dynamically estimating the variances, a computationally expensive procedure. Another problem is that Eq. (IV.20) may lead to negative bit allocations if the difference in variances between layers is large. This problem is correctable by forcing non-negative allocations in Eq. (IV.20) although the resulting allocations would not be optimal. Finally, Eq. (IV.20) does not take coding gain into account. Therefore, using R as the target bit allocation leads to bit allocations that are too low after taking VLC coding into account. One *ad hoc* fix is to replace R in the expression with

$$R' = R \cdot G, \tag{IV.23}$$

where G is the estimated coding gain expected from the entropy encoder.

More sophisticated algorithms using the rate-distortion concept are available. For example, "greedy" schemes allocate bits one at a time to the quantizer demonstrating the most distortion [78]. Other schemes apply Lagrange multipliers to arbitrary rate-distortion curves [80]. However, computational complexity and delay limit the feasibility of more advanced methods when dealing with real-time video.

2. Approximation of the 3-D Rate-Distortion Curve

The approach above assumes that distortion is additive in the operational coder and gives the same average distortion for each quantizer regardless of the relative perceptual importance of errors in each layer. The assumption of additive distortion implies that a decrease in rate requires a suitable decrease in all quantizer parameters to yield an optimal solution. Rate-distortion curves in the operational coder are not necessarily convex, so the above approach does not necessarily yield optimal results. An alternate, albeit heuristic, approach is to simplify the control problem by creating a simplified, operational rate-distortion curve.

An operational distortion curve is created by first plotting total bit rate and distortion (as measured by pSNR) separately in a three-dimensional space spanned by the set of candidate quantizers for a series of *motion* video sequences. This process captures the operational effect of the coder design, such as the quantizers and VLC coding as well as any interdependence between layers, on the rate-distortion relationship. The result is best described as a 4-D surface wherein both rate and distortion are functions of a triplet of quantizer parameters $\{q_1,q_2,q_3\}$. The first parameter represents the JPEG scaling factor while the remaining parameters represent the actual quantizer step sizes.

Next, the points representing the pSNR surface are sorted in ascending order and associated with their corresponding quantizer triplets. For those triplets producing approximately the same pSNR, only that point with the smallest bit rate is retained. The result is an implicit vector quantization of the operational 3-D rate-distortion surface. The dimensionality of the operational rate-distortion curve is therefore reduced to the 1-D curve covering the operational range of the coder as shown in Figure IV.23. Each point on the curve represents results from a single quantizer triplet. The corresponding quantizer triplets are plotted in Figure IV.24. The results indicate that an optimal rate control scheme does not necessarily increase/decrease each quantizer parameter in lockstep as would be expected if distortion in each layer were independent.

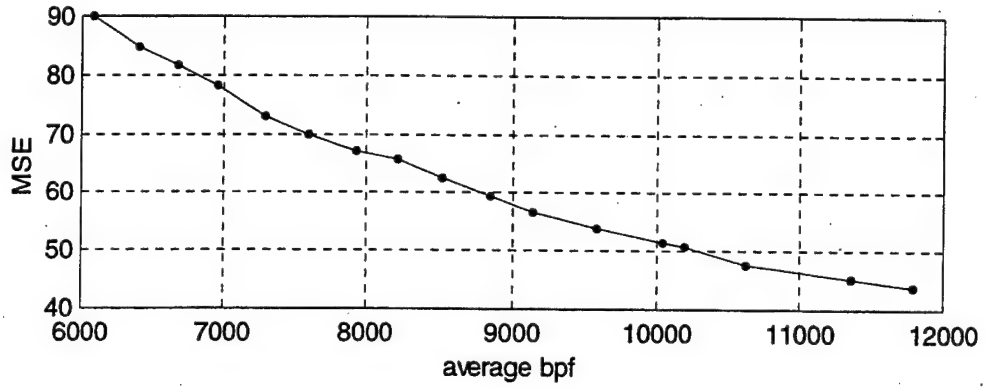

Figure IV.23: Operational Rate-Distortion Curve for Motion Video.

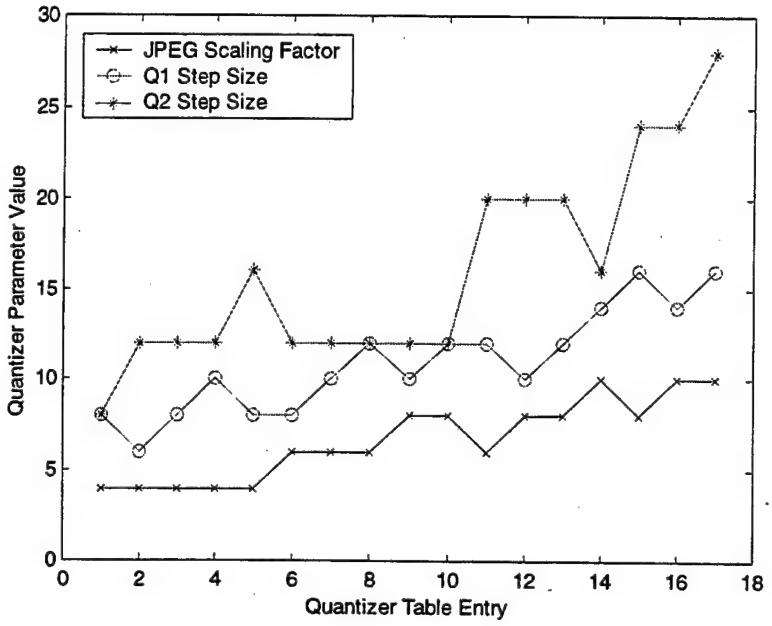

Figure IV.24: Quantizer Table Triplet Values for Motion Video.

Reducing the rate-distortion relationship to a suboptimal 1-D relationship provides a potential method for a simplified layered rate control scheme since the set of possible quantizer parameters is reduced to a more manageable set of suboptimal parameters. Considering each triplet as a suboptimal quantizer state, a feedback control

scheme manipulates the quantizers for each layer by selecting only entries from this set via table lookup. One possible method is considered next.

Using the operational rate-distortion curve, a control curve relating bits per frame to each suboptimal quantizer vector is created as shown in Figure IV.25. After linearizing the control curve over the operational range of the coder, the slope represents the average increment or decrement in bits per frame with a step change in the quantizer table. Dividing this quantity by the average number of macroblocks selected per frame in the test sequences yields the desired control parameter β,

$$\beta = \frac{\Delta B}{\Delta Q}\frac{1}{\overline{N}_{MB}} \tag{IV.24}$$

where $\overline{N}_{MB}$ represents the average number of macroblocks selected per frame and $\frac{\Delta B}{\Delta Q}$ is the slope of the control curve. In Figure IV.25, β was determined to be -11.0 bits/macroblock-step. The control parameter is then used to adjust the coder quantizer vector with each new frame as per the following scheme.

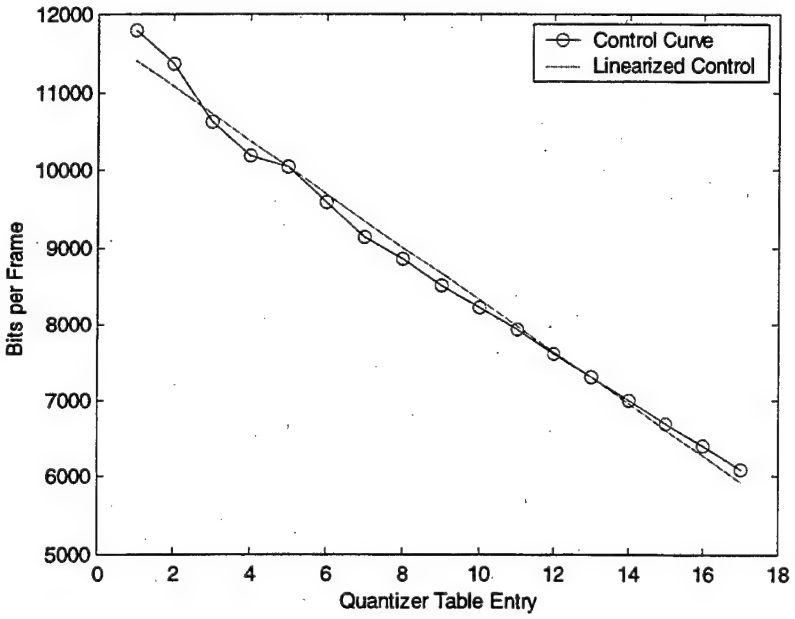

Figure IV.25: Operational Rate Control Curve.

At call setup, the average bit allocation per frame is set to

$$\overline{B} = \frac{R_{\text{target}}}{f}, \tag{IV.25}$$

where R_{target} is the channel bit rate and f is the frame rate. For each new frame I, we use the actual bit allocation from the last frame i - 1 to estimate the bit allocation error or deviation expected for the current frame i if the quantizer vector used in the last frame is not changed. Accounting for the change in the number of macroblocks selected between the last and current frames, the deviation expected is:

$$\Delta B_{\text{inter}} = \overline{B} - \left(\frac{N_{MB_i}}{N_{MB_{i-1}}} \right) B_{i-1}. \tag{IV.26}$$

The required change in the quantizer setting is calculated using the deviation ΔB_{inter}, the number of macroblocks selected for transmission in the current frame N_{MB_i}, and the control parameter:

$$\Delta Q_i = \left\lfloor \frac{\Delta B_{\text{inter}}}{N_{MB_i} \beta} \right\rfloor, \tag{IV.27}$$

where $\lfloor \ \rfloor$ is the fixed integer operator, which discards the decimal portion of the result. The result indicates that the quantizer setting from the last frame should be incremented or decremented by ΔQ_i. If the quantizer has reached the upper or lower limit of the table, the value is not changed.

Video traces for a rate controlled video sequence and a video sequence using only open-loop control are shown in Figure IV.26. Open loop control consists of selecting the quantizer setting that results in the bit rate closest to the one desired and then not changing the setting for the duration of the sequence. In each case, the target bit rate was 80 kbps. The results indicate that the frame-based rate controller maintains the local average closely and also smoothes the bit rate somewhat as measured by each sequence's variance. As presented in the next chapter, smoothing the bit rate increases multiplexer efficiency and reduces bandwidth requirements. The drawback of rate control is a slight

variation in frame-to-frame quality relative to open-loop control as shown in Figure IV.27. The statistics for each sequence are listed in Table IV.9. A variation of this approach was examined to increase the window used to predict the current deviation from just one frame as indicated in Eq. IV-28 to m frames to reduce bit variations. Offline coders look back m frames to calculate the deviation [81], but increasing the search window as in

$$\Delta B_{\text{inter}} = m\overline{B} - \sum_{j=1}^{m}\left(\frac{N_{MB_i}}{N_{MB_{i-j}}}\right)B_{i-j} \qquad \text{(IV.28)}$$

actually resulted in looser tracking in the sequences examined.

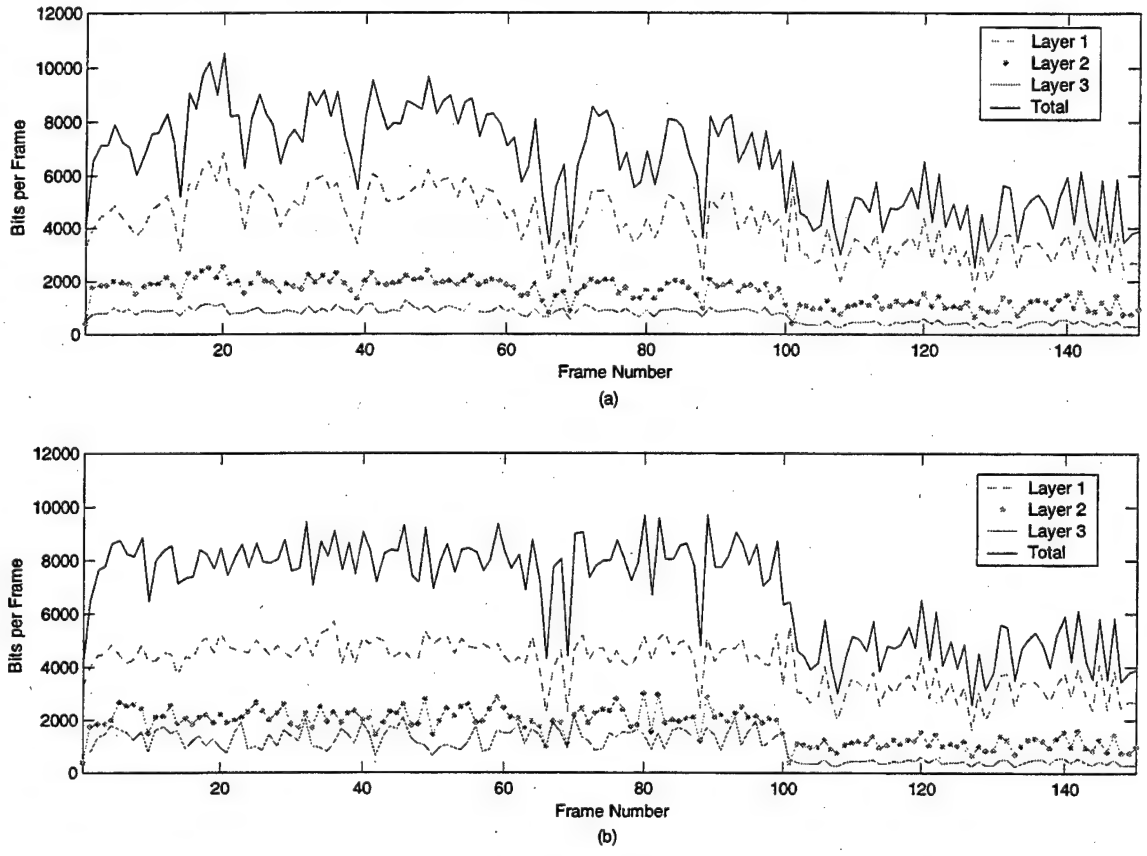

Figure IV.26: Bit Rate Traces for a) Controlled and b) Uncontrolled Video Sequences.

Changing the quantizer only at the beginning of each frame may provide insufficient granularity to adequately suppress deviations from the desired bit rate. In this case, a more desirable approach is to examine the quantizer vector each macroblock and make changes as required to control the target bit distribution among the macroblocks. However, this approach is more complex than frame-based control and may cause quality variations throughout the frame during high activity periods. One simple scheme is to distribute the average bit allocation for each frame evenly among all the selected macroblocks in a similar manner to the Telenor rate control scheme [55]. Given that N_{MB_i} macroblocks are selected in the current frame and an average bit allocation of $\overline{B}$ bits is used, each macroblock receives $\overline{B}/N_{MB_i}$ bits.

Controlling bit rate at the macroblock level is performed as follows. At call setup, average bit allocation per frame is set to

$$\overline{B} = \frac{R_{\text{target}}}{f}, \tag{IV.29}$$

where R_{target} is the channel bit rate and f is the frame rate. For the first macroblock of the new frame i, we calculate the expected deviation in the bits allocated to the current frame if the quantizer setting from the last frame is not changed as above and apportion this deviation over the number of macroblocks selected. This value is used to determine the change required in the quantizer setting for the first macroblock:

$$\begin{aligned} \Delta B_{\text{inter}} &= \overline{B} - \left(\frac{N_{MB_i}}{N_{MB_{i-1}}} \right) B_{i-1}, \\ \Delta Q_{i,1} &= \left\lfloor \frac{\Delta B_{\text{inter}}}{N_{MB_i} \beta} \right\rfloor. \end{aligned} \tag{IV.30}$$

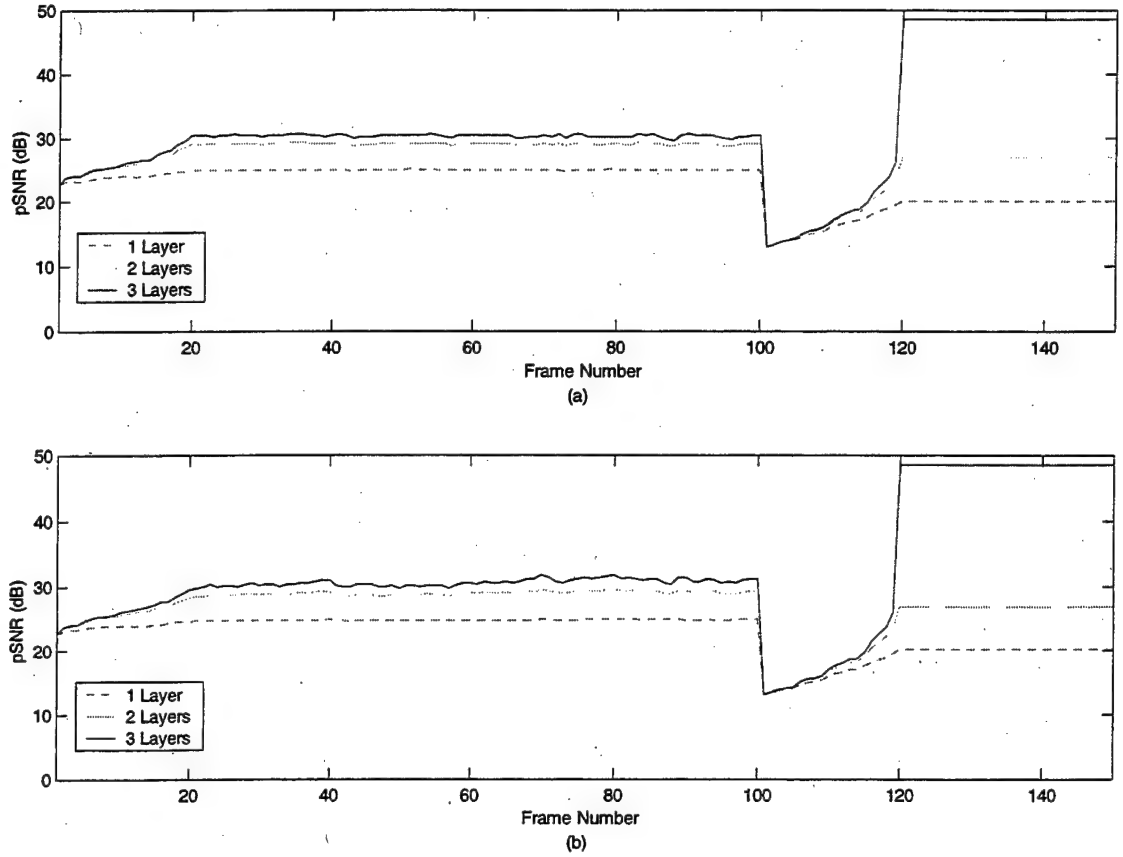

Figure IV.27: pSNR Variation for a) Controlled and b) Uncontrolled Video Sequences.

Parameter	With Rate Control	Without Rate Control
Mean Bit Rate (bpf)	7998	7454
Bit Rate STD (bpf)	942	1362
Mean pSNR (dB)	29.83	29.51
pSNR STD (dB)	1.92	1.74

Table IV.9: Rate Controlled and Uncontrolled Sequence Statistics.

For each remaining macroblock j, $j = 2$ to N_{MB}, we calculate the deviation between the bits allocated so far within the frame and the target linear distribution. Assuming that the number of bits allocated so far within the current frame is $B_{i,j-1}$, and given that the target bit allocation per macroblock indicated by Eq. (IV.30), the deviation at macroblock j is:

$$\Delta B_{\text{intra}} = \frac{(j-1)\overline{B}}{N_{MB_i}} - B_{i,j-1}. \tag{IV.31}$$

This deviation is then used to set the quantizer parameter for the current macroblock:

$$\Delta Q_{i,j} = \left\lfloor \frac{\Delta B_{\text{intra}}}{\beta} \right\rfloor. \tag{IV.32}$$

One possible objection to rate control at the macroblock level using the scheme above is that the linear bit allocation across the selected macroblocks takes into account neither the level of activity within each macroblock nor the perceptual importance of individual macroblocks. Therefore, the linear approach can be generalized by introducing a weighting factor W_j for each macroblock that represents the relative proportion of bit allocation to be assigned to that macroblock:

$$B_{i,j} = W_j B_i. \tag{IV.33}$$

The only constraint placed on W_j is that all weights sum to 1 to achieve

$$\sum_j B_{i,j} = \sum_j W_j B_i = B_i. \tag{IV.34}$$

The linear assignment scheme, with $W_j = 1/ N_{MB_i}$ obviously meets this condition. Two approaches provide a means to tailor bit activity to macroblock activity level. First, macroblock selection rate provides a heuristic indication of motion within the current scene. Given the set of macroblocks selected for the current frame, each macroblock's past selection history can be used to determine a selection probability p_j relative to the current set. Such a selection probability provides a convenient measure of motion. Those blocks that are selected more often tend to lie in regions of greater motion. Therefore, an appropriate weighting factor that emphasizes regions of greater motion is to set

$$W_j = p_j . \tag{IV.35}$$

However, the coder must refresh selection counts after every scene change to avoid biasing the motion detection. Another approach is to weight the bit allocation by the variance exhibited by each macroblock, thereby allocating more bits to macroblocks with higher variance. A similar approach is followed in [77]. Using the rate distortion allocation scheme outlined above, a weighting factor based on variance is

$$W_j = \frac{B_{i,j}}{B_i} = \frac{B_i + \log_2 \frac{\sigma_j}{\rho}}{B_i} = 1 + \frac{1}{B_i} \log_2 \frac{\sigma_j}{\rho}, \tag{IV.36}$$

where B_i is the current frame bit allocation, $B_{i,j}$ is the allocation for the jth selected macroblock, and σ_j is the variance of the coefficients in the jth macroblock. The only drawbacks to this scheme are that weights may be negative and macroblock variance must be tracked, which increases computational overhead.

Continuing this approach with static video produces interesting results. As shown in Figure IV.28, the operational rate-distortion curve is relatively flat over a wide range of bit rates. Since the coder's operational range falls into this region, rate control as described above is not possible since all of the quantizer states produce the same level of quality. However, rate control is not a distinct requirement for static sequences. Since macroblocks are only transmitted due to aging, bit rates for static sequences are considerably less than those observed in motion video sequences. Accordingly, open loop rate control is adequate for static sequences. The quantizers are preset for static sequences to the quantizer triplet that yields the lowest bit rate in the flat distortion region and fixed for the duration of the sequence.

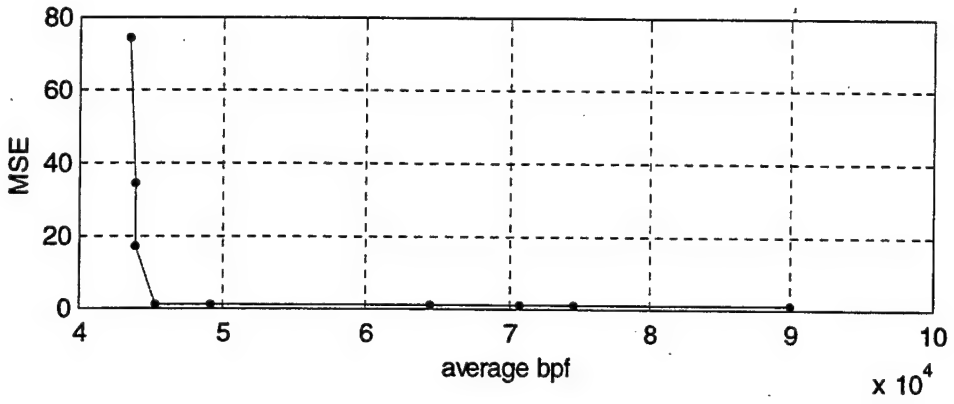

Figure IV.28: Operational Rate-Distortion Curve for Static Sequences.

The clear implication of rate control is that any change in the quantizer setting must be communicated to the decoder. Although the operational control curve shown in Figure IV.25 reduces the amount of data used to describe each quantizer state, transmitting the quantizer setting consumes bandwidth, and update frequency should be minimized. Therefore, at a minimum, the current quantizer vector must be transmitted with the frame header using frame-based rate control and with each macroblock using macroblock-based rate control. In either case, using a VLC code to communicate only the change in quantizer setting, as in differential pulse coding, can further reduce overhead. However, the minimal approach directly conflicts with the need for robust coding. If the frame header is damaged, the quantizer settings for that frame are lost. Differential coding creates a liability unless some facility is made for refreshing the quantizer state after any interruption due to lost cells. To ensure that each GOB is independently decodable for robustness, the following compromises are possible. For frame-based rate control, the quantizer setting, in the form of the lookup table index, is included in every GOB header. For macroblock-based rate control, the quantizer setting is coded differentially between macroblocks within the GOB and refreshed every GOB. Differential coding within the macroblock poses no liability since a dropped cell interrupts decoding until the decoder resynchronizes with the next GOB header.

This chapter introduced a new layered coder design motivated by the need to provide a flexible video delivery scheme for greater robustness over heterogeneous networks. Attention was focused on those elements required to promote the *effectiveness* of layered coding. In general, the coder uses the fast Haar transform to decompose selected macroblocks into subbands, and then subbands are allocated to layers based on their relative perceptual importance. Specifically, a generalized layering scheme was devised for motion video that allows creation of an arbitrary layering scheme as a function of video content as evidenced by subband variance. However, a common layering scheme for motion video and static presentation slides is impractical since each attaches a different perceptual relevance to the various subbands. Therefore, different layering schemes are employed for each type of video content; the coder picks the appropriate scheme dynamically within the video sequence.

A final issue examined was that of rate control for the layered video sequence. Since subbands are essentially layered by common variance, each layer employs a different quantization scheme. Rate control via traditional rate-distortion techniques is complicated by the increased dimensionality of the layered coder's rate-distortion surface and the possible inter-dependence among quantizers. Rate control is simplified by selecting a suboptimal set of quantizer vectors, where each vector consists of step size for each quantizer, thereby effectively reducing the operational rate-distortion curve to a 1-D relationship. Rate control, either at the frame level or macroblock level, is implemented via a simple table lookup.

V. TRAFFIC SMOOTHING

The previous chapter presented a new scheme for preparing a video sequence for transmission over the network by coding the sequence as a hierarchical series of layers. The next chapter exploits the relative perceptual importance of each layer through priority-based scheduling. However, the manner in which the layers are transmitted to the network, i.e. the statistical characteristics of each cell flow, plays a role in determining the resources each switch must commit to the sender to guarantee that sender's required QoS. In general, the more random the cell flow, the more resources, such as bandwidth, must be committed. Consequently, by manipulating the statistical characteristics of each traffic flow prior to the network, the network's capacity for carrying traffic is enhanced, which is particularly desirable for low-bit-rate networks.

This chapter examines the concept of traffic smoothing for layered video traffic as a means for increasing transmission robustness by increasing queuing efficiency. The chapter starts by discussing the concept and application of traffic smoothing. Next, the psuedo-histogram traffic model proposed by Skelly et al. for VBR video is presented [14]. The psuedo-histogram has the advantage of capturing the effect of frame-by-frame smoothing on queue behavior. Details on determining model parameters and analytical techniques for *D/D/1/K* queues are presented including a simple technique for rate-controlled video. Finally, an integrated scheme is proposed for traffic smoothing of layered video traffic at various time scales: frame level, layer level, and cell level. The issue of where to apply traffic smoothing for the single VCC and multiple VCC cases is examined along with the issue of mitigating delay added by frame-by-frame smoothing.

A. INTRODUCTION

One of the functions of ATM traffic management is call acceptance, which ensures that sufficient network resources exist prior to accepting a new connection with specified QoS requirements. The requisite resource allocation as a function of the required QoS depends on statistical properties of the connection's traffic flow. The

requisite allocation may also depend on the properties of other connections currently within the network. Each new connection characterizes its anticipated traffic properties via a set of descriptors that depend on the type of service required [28]. Possible traffic descriptors include peak cell rate (PCR), a sustainable cell rate (SCR), and the maximum burst size (MBS). The network layer then uses these traffic descriptors and the current network state to determine whether to admit the call. If the call is admitted, a traffic contract is formed between the connection and the network. The connection agrees to abide by the traffic descriptors and the network agrees to allocate resources such that the connection's QoS is maintained.

Assuming that the VCC traverses sequential queues, QoS is guaranteed by ensuring that sufficient channel allocation exists at each queue such that the QoS parameters are maintained. Focusing on an individual queue, the required channel allocation depends on the arrival process, the QoS required, and the service process. For ATM networks, service is deterministic. However, the service rate depends on the required QoS and the arrival process. For a given QoS and a given arrival process, the goal is to minimize the service rate required.

Since QoS is usually fixed for each particular traffic type, the arrival process weighs heavily in the channel allocation. The traffic flow within each connection may be viewed as a random process. In general, the channel allocation to that traffic flow depends on the relative uncertainty or random variation in its arrival process at a particular queue. In particular, the greater the uncertainty in a traffic source's arrival process, the greater the bandwidth required to meet the desired QoS. For example, CBR traffic is completely characterized by its peak cell rate alone. By definition, the instantaneous arrival rate for VBR traffic is time varying although the average rate is fixed[12]. A simple method for characterizing the variation in the arrival rate is the ratio of PCR to average cell rate [82]. This ratio represents the burstiness of the source; a higher ratio denotes a burstier source. For a CBR source, this ratio is one. Alternately, the

[12] Otherwise an ATM network would not be able to ensure QoS for the duration of the connection.

burstiness of a VBR source can also be expressed in terms of the variance of cell interarrival times [82].

The problem of bandwidth allocation for a bursty source may be viewed from the perspective of a deterministic ATM queue. A connection is guaranteed to lose no cells if the service rate exceeds the arrival rate. With a bursty source, selecting the service rate equal to source's PCR ensures that no cells are lost. However, the channel is underutilized with this allocation. Selecting the service rate equal to the average cell rate fully utilizes the channel but leads to a large amount of cell loss. Given an acceptable CLR, the appropriate service rate lies between the PCR and the average cell rate, which implies that a certain amount of underutilization must be tolerated to achieve the desired QoS. Of course, this exact characteristic provides the basis for statistical multiplexing since the aggregate multiplexed source is considerably less bursty than each individual source.

Given that uncertainty in the arrival process increases bandwidth requirements, altering a connection's traffic characteristics through traffic shaping is desirable to increase the number of connections that may be serviced with a given amount of bandwidth. Alternately, traffic smoothing increases robustness during periods of congestion since leveling out bursts tends to reduce the probability of buffer overflows. Both considerations are especially important given the low bandwidth VTC scenario presented here. Traffic shaping may be further differentiated into the functions of traffic smoothing and traffic policing. Traffic smoothing attempts to reduce or control burstiness either at the application level or at some point prior to entry into the network. Traffic policing monitors a connection's traffic parameters and takes action to correct deviations. For example, Usage Parameter Control (UPC) in ATM monitors each connection to ensure that its traffic conforms to the traffic contract [18][28]. Non-compliant cells are tagged and may be dropped later in the network to avoid impacting the QoS guaranteed to other connections. The two functions are not totally unrelated; controlling burstiness, perhaps at the application level, may be viewed as a form of self-

imposed traffic policing. Here, attention is focused only on the application of traffic smoothing on video traffic.

The first logical place to implement traffic smoothing is at the application level through rate control. Rate control as presented in the last chapter represents a type of self-imposed traffic policing; the rate controller attempts to maintain some traffic statistic at a fixed level through control over the quantizer setting. However, rate control provides an obvious mechanism for traffic smoothing. Forcing transmitted video to a constant bit rate completely removes the burstiness inherent in video traffic, but at the cost of potentially wide variations in quality from frame to frame. A less severe tradeoff is to settle for a constant mean bit rate which is the approach taken in Figure IV.26. In this case, quality variations between successive frames are less noticeable, and the level of burstiness is decreased as indicated by the drop in bit rate variance (see Table IV.9). Before rate control, the burstiness factor is 1.41; after imposition of rate control, the burstiness factor drops to 1.21. Of course, controlling only the mean bit rate does not guarantee any particular degree of smoothness. With proper design, a rate control scheme should be able to achieve an arbitrary level of smoothness that is bound only by the permissible coding delay.

A more general method for smoothing a traffic flow prior to entry into the network is the leaky bucket scheme proposed for network access control [83][84]. Access control ensures that a traffic source does not exceed its traffic parameters agreed to as part of the traffic contract. The scheme is illustrated in Figure V.1. The basic idea is that the leaky bucket mechanism controls access to the network. ATM cells arriving at the leaky bucket must obtain a token from a token pool to enter the network. Tokens are generated at a constant rate r and placed in the token pool. Additionally, there is a maximum limit on the number of tokens in the token pool at any time, and tokens arriving after the token pool is full are discarded. The token pool is sized to control the maximum burst length from the source, i.e., the maximum number of cells that can be transmitted back-to-back. Restricting the number of tokens controls the burstiness of the source while the token rate dictates the average cell rate. If a cell arrives and a token is

not available, three courses of action are available. The cell could be discarded; the cell could be buffered until a token becomes available; or the cell could be tagged as non-compliant and transmitted. The cumulative affect of buffering and manipulating the token rate allows considerable flexibility in altering traffic statistics. However, buffering introduces delays in the forward transmission path, and the gain offered by smoothing must be weighed against the added delay.

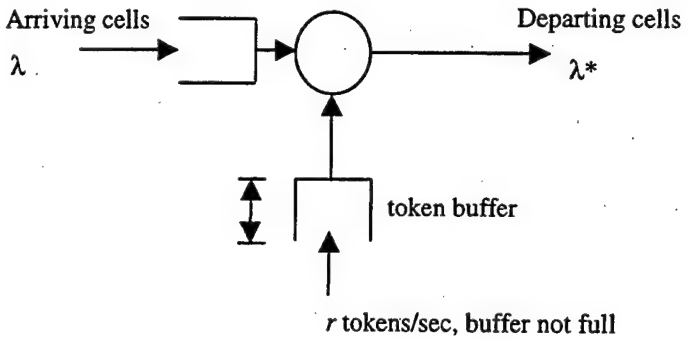

Figure V.1: Leaky Bucket Access Mechanism.

While originally conceived as an access control mechanism, the leaky bucket scheme controls by smoothing the traffic flow. However, smoothing is performed for the purpose of ensuring compliance with the traffic contract. The approach may be generalized for smoothing at other points prior to network entry, such as at the application level prior to the AAL or within the AAL prior to the ATM layer. In either case, tokens are used to permit transfer of PDUs instead of ATM cells. This offers another avenue for smoothing video traffic prior to network entry. For example, a CBR type smoothing can be implemented by setting the token rate r proportional to the channel rate and setting the token pool size to one. Then, arriving PDUs are buffered and transmitted to the next lower layer at the token rate, maximizing smoothness but potentially increasing the transmission delay.

Given the impact of traffic statistics on queuing efficiency, characterizing VBR video traffic sources via stochastic models plays an important role in network performance analysis. In particular, traffic models provide a powerful tool for analyzing

the impact of the arrival process on queue behavior through either simulations or analytical analysis. For example, traffic models can provide insight into determining appropriate tradeoffs between buffer depth and service rate to achieve a desired QoS. For a traffic model to be useful, the model should perform two functions. First, the model must accurately represent traffic statistics, namely the first and second moments and the covariance function. Second, to evaluate QoS metrics such as cell delay and cell loss and to validate simulation results, the traffic model should extend to some form of analytical queuing analysis. Meeting both of these goals is a non-trivial task.

B. VIDEO TRAFFIC MODELING

This section presents three VBR video traffic models as background for traffic simulations conducted in later sections and to motivate, in part, the smoothing mechanism presented in the next section. The autoregressive models proposed by Maglaris et al. [86] and Sen et al. [88] are interrelated and have been used to model VTC video traffic [27]. The histogram-based video traffic model proposed by Skelly et al. [14] is notable in that it captures the effect of smoothing video traffic on a frame by frame basis and provides particularly versatile queuing analysis techniques.

Modeling VBR traffic requires capturing the interdependence between coder design and video activity level that influence the video stream's arrival process. Important factors with regard to the coder are the compression scheme employed, particularly in the distribution of I- and P-frames, and the presence of rate control. Video activity influences the compression gain through the level of scene activity or motion and the periodicity of scene changes. Video traffic models attempt to accurately capture the first and second moment statistics of the traffic source along with its covariance function. A useful traffic model also incorporates queuing analysis techniques that allow calculation of QoS metrics, such as cell delay and cell loss rate, to validate simulation results. Another desirable trait is low computational complexity.

1. Autoregressive Models

A representative video trace, in bits/pixel, is shown in Figure V.2 for a rate-controlled "talking head" scene typically found in VTC. Such sequences usually are characterized by a roughly Gaussian shaped bit rate histogram and an exponentially decaying autocorrelation function. On the strength of these observations, VBR traffic models based on a first order autoregressive processes have been proposed by Maglaris et al. [86] and Heyman et al. [87]. Using a first order autoregressive model, the variation in bit rate is expressed as

$$\lambda(n) = a\lambda(n-1) + bw(n) \tag{V.1}$$

where $w(n)$ is Gaussian white noise with unit variance but a non-zero mean. The parameters in Eq. (V.1) are determined using the first and second-order statistics measured from the video sequence along with the estimated autocorrelation function.

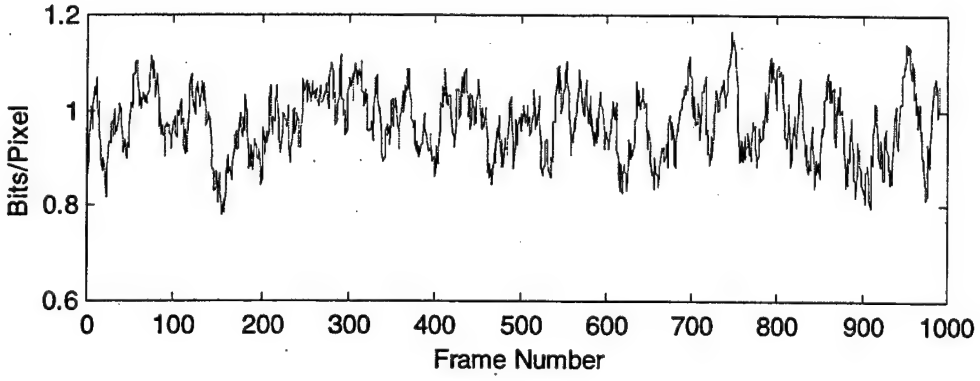

Figure V.2: Video Trace for a Low Activity Sequence.

Although a first order autoregressive process captures the effect of bit rate variation, these models provide little insight into queuing behavior. Sen et al. [88] has proposed a model for N multiplexed video sources that can be applied in queuing analysis. The model represents the aggregate video sequence as the output of M multiplexed identical, two-state Markov chains, or minisources, where $M >> N$. Each minisource alternates between an off-state and an active state as shown in Figure V.3. When multiplexed, the minisources yield an equivalent $(M + 1)$-state Markov chain

wherein each state transmits at a fixed multiple of R cells/second. Using 20 or more minisources per video source reduces the affect of quantization. The model's parameters, α, β, and R, are determined from the first and second moments as well as the autocorrelation function for a single video source; all video sources are assumed to have the same statistical characteristics. Given the model parameters, cell loss probability and buffer occupancy statistics are determined through fluid-flow analysis [27]. A shortcoming of the minisource model is the inability to model an arbitrary bit rate histogram since bit rate follows a binomial distribution [14].

While both of the above models do a good job of characterizing bit rate variations within a scene, no attempt is made to capture the effect of scene changes. Given the behavior of motion-compensated video coders, aperiodic bit rate peaks are expected due to scene changes since, following a scene change, most macroblocks are intracoded due to a lack of a suitable reference in the last frame[13].

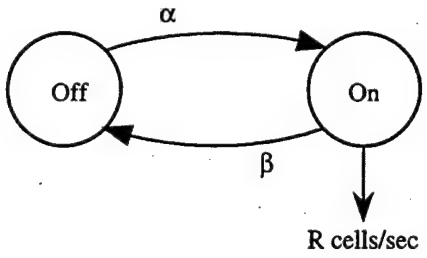

Figure V.3: Minisource Video Model.

2. Histogram-based Traffic Modeling

The histogram-based video traffic model proposed by Skelly et al. [14] represents an intermediate approach between autoregressive modeling and self-similar traffic models. The premise of the model is very simple: quantize the arrival rates and then approximate the video sequence by its quantized version. Motivation for the model stems from the need to smooth the video traffic flow. Dixit and Skelly, in an earlier work [89],

[13] For an analogous reason, periodic bit rate peaks occur in MPEG-encoded sequences due to the GOP structure.

demonstrated the relation between traffic smoothing and ATM multiplexer performance. Given a buffered, compressed video frame, the resulting ATM cells could be transmitted in several manners. For example, the cells could be transmitted at the peak available channel rate until the buffer is emptied. The resulting traffic is very bursty since the video coder transmits at a high rate for a brief period and then falls idle for the rest of the frame. The problem with this approach is that when several sources are multiplexed, any correlation between the burst periods tends to increase cell loss dramatically.

Dixit and Skelly [89] instead proposed to transmit the buffered cells randomly over the entire frame interval as a Poisson stream to the ATM multiplexer. Skelly et al. [14] combined this smoothing scheme with the quantized video traffic model described above. Each quantized level represents a single frame, and cells from each quantized level are transmitted as a Poisson stream to the multiplexer over one frame interval. Assuming that transitions between levels may occur every frame and that the transitions are memoryless, the resulting traffic model is a discrete-time multi-state Markov-modulated Poisson process (MMPP) as shown in Figure V.4 (some transitions are removed for clarity). The Markov chain serves to modulate the underlying Poisson-smoothed arrival process, where each state i corresponds to a Poisson process whose arrival rate λ_i matches the size of the compressed frame in bits for that state. Shroff [15] later expanded the MMPP model into the generalized histogram model, also known as a Markov-modulated rate process (MMRP), which incorporates arrival processes other than Poisson [15]. In particular, Shroff demonstrated that the maximum queuing efficiency in ATM multiplexers is achieved by smoothing deterministically, i.e., by transmitting cells at equal intervals throughout the frame interval. The result resembles a modulated CBR process with a new rate every frame.

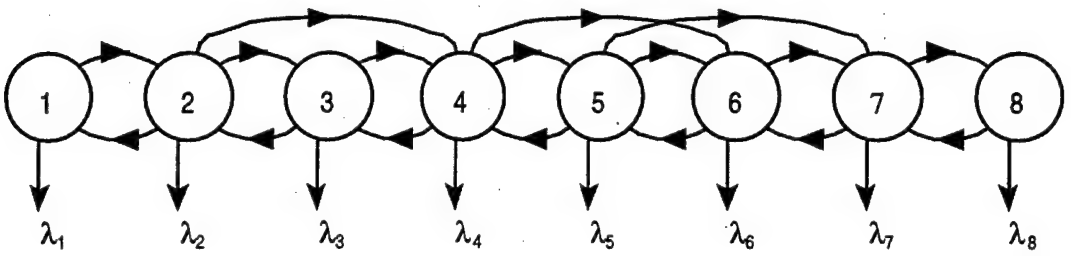

Figure V.4: Markov-modulated Poisson Process (MMPP).

3. Determining Model Parameters

The histogram model parameters consist of the MMRP state probabilities, the state transition probabilities, and the state arrival rates and are estimated from the video sequence in the following manner. The video sequence is uniformly quantized into n bins, where each bin represents a single state. The quantized arrival rates λ_i represent the arrival rates for their respective states. Next, transition probabilities between states are measured directly from the quantized sequence yielding the state transition matrix P. The steady state distribution is given by

$$\pi = [\pi_1 \quad \cdots \quad \pi_n], \tag{V.2}$$

where π_i is the steady-state probabilities for state i. The state probabilities can be determined by solving the eigenequation:

$$\pi = P\pi\,. \tag{V.3}$$

Alternately, π is the eigenvector of P whose corresponding eigenvalue is 1 [49]. Since the rate of the modulating process is much slower than the modulated process, an equivalent continuous-time Markov process is determined from [27]

$$M = f(P - I)\,, \tag{V.4}$$

where f is the frame rate, and M is the infinitesimal generating function representing transition rates from each state.

Once the model parameters have been determined, one check of the model's fitness is to compare the model's first and second moments and autocorrelation function to those of the actual sequence. For the model, the mean is given by:

$$E[\lambda(n)] = \sum_{i=1}^{n} \pi_i \lambda_i \,. \tag{V.5}$$

The autocorrelation function is given by [27]:

$$E[\lambda(n)\lambda(n+l)] = \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j P[\lambda(n+l) = \lambda_j | \lambda(n) = \lambda_i] P[\lambda(n) = \lambda_i]. \tag{V.6}$$

Since the histogram model approximates the actual histogram of the given video sequence, the model is able to support a wide range of video activity and compression schemes. For example, while the MMRP model does not explicitly model scene changes, the peaks in bit rate resulting from scene changes are implicitly captured in the higher states. Skelly et al. [14] presented results from 10 second JPEG encoded sequences taken from "Star Wars". Compared to the original sequences, the eight-bin model predicts a slightly higher mean bitrate and provides a good match for the autocorrelation function over a range of four seconds (96 frames). While increasing the resolution of the histogram did not dramatically change the approximation, employing less than eight bins resulted in a poor approximation. With rate-controlled video segments, satisfactory results have been reported using as few as six states [90].

Given the histogram parameters for a single source, an equivalent histogram for N homogenous sources may be obtained through $N-1$ convolutions [91]:

$$\pi^A = \pi * \pi * \cdots * \pi \,. \tag{V.7}$$

The state arrival rates are given by

$$\lambda_i^A = N\lambda_1 + (i-1)\Delta\lambda, \;\; \Delta\lambda = \lambda_2 - \lambda_1, \; i = 1,2,\ldots,2N-1. \tag{V.8}$$

For heterogeneous sources, the process is slightly more difficult and the equivalent histogram must be resolved one source at a time. Given two non-equivalent histograms, the joint histogram may be written as a two-dimensional Markov chain with N^2 states [27]. The probability for state (m,n) is given by

$$\pi_{m,n} = \pi_m \pi_n \tag{V.9}$$

and the aggregate arrival rate by

$$\lambda_{m,n} = \lambda_m + \lambda_n, \tag{V.10}$$

where the indices *m,n* refer to the corresponding states in the original Markov chains. The result can be converted back into a one-dimensional histogram by renumbering the states in order of increasing arrival rate. Compared to the homogeneous case, the size of the aggregate histogram grows much more rapidly although coalescing states or deleting highly improbable states, in comparison to the simulation length, can possibly reduce the size.

4. Queuing Analysis

Cell loss analysis proceeds by invoking a quasi-static behavior for the MMRP model and assuming that the rate of the modulating process, the Markov chain, is far slower than the rate of the modulated process, the state arrival rate. With this assumption, the queue is expected to reach equilibrium rapidly compared to the time interval between frames, and each state may be treated as an independent source. The probability that the buffer contains *n* cells is given by [27]:

$$P[N=n]=\sum_{i=1}^{n} P[N=n|\lambda=\lambda_i]\pi_i\,, \qquad \text{(V.11)}$$

where π_i are the state probabilities, and $P[N=n|\lambda=\lambda_i]$ is the probability that the buffer contains *n* cells given the arrival rate λ_i. From Eq. (V.11), the buffer distribution for each individual state depends on the arrival process to the buffer, which in turn depends on the smoothing mechanism and the type of service granted. Given that ATM uses fixed-length cells, service is usually deterministic. Although the original histogram model used Poisson smoothing, Shroff [15] has demonstrated that deterministic smoothing yields better queuing performance. Therefore, further discussion is limited to only *D/D/1/K* queuing systems. Equation (V.12) indicates that the transition rates between states, and by extension the shape of the autocorrelation function as given by Eq. (V.6), play no role in determining the buffer occupancy distribution as would be expected if self-similarity is a significant factor. Indeed, Skelly's [14] results indicate that accurately capturing the autocorrelation function plays a greater role in modeling buffer distributions than the actual shape of the autocorrelation function.

Although buffer distribution is of interest, analyzing cell loss is more important in determining appropriate buffer depths. The system loss probability, assuming that states are independent, is given by [27]

$$P_L = \frac{1}{E[\lambda]} \sum_{i=1}^{n} P_{L_i} \lambda_i \pi_i \,, \tag{V.12}$$

where $E[\lambda]$ is given by Eq. (V.5), π_i are the state probabilities, λ_i are the arrival rates, and P_{L_i} are the loss probabilities for that state. Equation (V.12) represents the aggregate loss rate as the sum of cells lost from each state over a long interval, weighted over each of n states, divided by the expected number of arrivals. The individual loss rates in Eq. (V.12) depend on the queuing system being evaluated. For a *D/D/1/K* queuing system, assuming a very long sojourn time T for each state, allows a simple approximation for loss rate [15]. If the arrival rate is less than the service rate, no cells will be lost since an arriving cell finds the server idle or servicing a cell. If the service rate is less than the arrival rate, cells not serviced during the sojourn time or buffered are lost. The loss probability in this case is given by:

$$\begin{aligned} P_L &= \lim_{T \to \infty} \frac{\lambda T - \mu T - K}{\lambda T} \\ &= 1 - \frac{1}{\rho}, \quad \rho \equiv \frac{\lambda}{\mu}, \end{aligned} \tag{V.13}$$

where T is the sojourn time, λ is the arrival rate, and μ is the service rate. Considering both scenarios, the loss rate for the ith state for determistic arrivals and determinisitc service is:

$$P_{L_i} = \begin{cases} 0, & \rho_i \equiv \dfrac{\lambda_i}{\mu} < 1, \\ 1 - \dfrac{1}{\rho_i}, & \rho_i > 1. \end{cases} \tag{V.14}$$

Substituting the result from Eq. (V.14) for each state into Eq. (V.12) gives the system loss probability.

For *D/D/1/K* systems, Eq. (V.14) indicates the counterintuitive result that cell loss probability is independent of queue size K. However, cell loss behavior demonstrates

two distinct patterns dependent on buffer size, the cell region and the burst region, as shown in Figure V.5 [27]. In the cell region, cell loss drops rapidly with buffer size, and cell losses are confined to individual cells. This region is modeled well by Eq. (V.12). In the burst region, cell loss drops at a slower but exponential rate with buffer size; cell losses occur in bursts in this region, a behavior not captured by the histogram model. Equation (V.14) indicates both regions coalesce into a constant value for *D/D/1/K* systems. However, simulations show that these systems lie instead in the burst region [27].

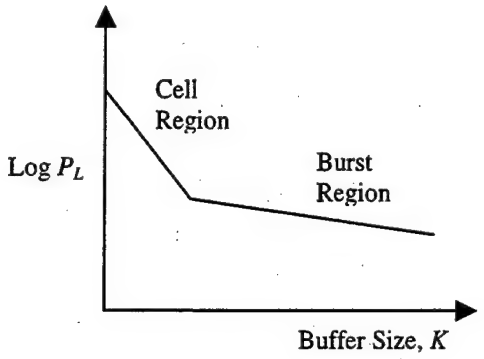

Figure V.5: Cell and Burst Regions for Cell Loss.

Shroff offers an ad hoc technique for estimating cell loss probability using MMRP models by incorporating fluid level analysis to capture behavior in the burst region [15]. In the cell region, loss is calculated using Eq. (V.12) with an appropriate expression for P_{Li}. In the burst region, fluid level analysis is used to predict the exponential relationship with queue size in the form,

$$P(x > K) = Ae^{\delta K} \tag{V.15}$$

where δ is dominant eigenvalue from the fluid level representation of the system. Using the infinitesimal generating function for the histogram model, δ is the least negative eigenvalue of the array $D^{-1}M$, where D is given by:

$$D = diag[\lambda_i - \mu]. \tag{V.16}$$

The constant A in Eq. (V.15) is determined by piecing the cell region and burst region curves together at the cutoff point K_0 where both curves have equal slopes. Then the

constant A is a function of the cutoff buffer size and the cell region loss probability at that buffer size,

$$A = P(x > K_0)\Big|_{cell\ region} \times e^{-\delta K_0}. \qquad \text{(V.17)}$$

Together, Eq. (V.12) and Eq. (V.15) provide a complete description of the cell loss behavior with queue size. A MMRP system with deterministic arrivals represents a special case since the queue is always in the burst region. The cell loss probability is determined by correcting Eq. (V.12) directly by the factor $e^{K\delta}$.

For multiplexed sources, cell loss probability is determined by applying the above techniques to the equivalent histogram resulting from numerical convolution of the individual histograms. Shroff's technique extends easily in the case of multiplexed homogeneous sources but becomes more difficult with heterogeneous sources [92].

5. Application

In the next section and the next chapter, a MMRP model is used to represent a deterministically smoothed layered video traffic source. The model is used both as a traffic source in OPNET simulations and as an analytical model for queuing calculations. Model parameters were derived from the rate-controlled sequence shown in Chapter IV. The actual parameters are given in Appendix B.

C. SMOOTHING LAYERED VIDEO TRAFFIC

Traffic smoothing improves multiplexer performance; the implied benefits are a degree of bandwidth conservation, which permits the network to guarantee QoS for a given level of traffic with less bandwidth. This is particularly desirable for the low bit rate network envisioned in Chapter II. While smoothing has been discussed previously, coverage has focused on network-level traffic shaping for both traffic policing and improving multiplexer performance. In this section, we propose a new smoothing scheme targeting layered video that is notable in two ways. First, we focus on developing a practical smoothing mechanism implemented at the sender prior to the ATM layer. The goal is to avoid manipulating traffic streams at the ATM layer since

maintaining a separation between network and client layer functionality is desirable to preserve network interoperability. Second, the smoothing mechanism covers three time scales: frame level, layer level, and cell level. The former is considered briefly while the latter two are the main focus of this section.

Based on previous discussion, rate control provides frame-level smoothing by limiting variations from the target bit rate. This type of smoothing is obtained essentially as a byproduct since rate control is a necessary component to ensuring compliance with the traffic contract in ATM networks. As shown in Figure IV.26 and using the values given in Table IV.9, the rate control mechanism discussed in Chapter IV produces an approximate 16% decrease in burstiness.

1. Cell Level Traffic Smoothing

While rate control smoothes variations in bit rate over multiple frames, a more explicit approach is to smooth at the cell level by controlling interarrival times to the ATM multiplexer. As discussed in the last section, this exact concern partially motivated Skelly's [14] histogram traffic model. Following Skelly's approach of smoothing individual frames, we propose an analogous smoothing scheme implemented via a leaky bucket type mechanism. The basic approach is shown in Figure V.6. Smoothing proceeds by modulating the arrival rate into the network for each individual frame. Each compressed frame is buffered prior to transmission into the network, and portions of the compressed frame, termed transmission units for now, are released for transmission whenever a token is available. Tokens are generated at a fixed rate r and only a single token is available at a time. The combined effect is to deterministically smooth the flow of transmission units by releasing them for transmission at intervals of $1/r$ seconds. The token rate r is evaluated anew each frame and is set to the arrival rate for the current frame as measured in transmission units per second. In this manner, the token rate is assigned a value sufficient to ensure that the entire frame is transmitted during a single frame interval. For example, if a transmission unit consists of 300 bits and the current compressed frame size is 6000 bits, the token rate must be set to $20f$ tokens per second, where f is the frame rate. Since this scheme occurs downstream from the video coder,

rate control is not explicitly a part of the smoother. However, the benefit of rate control appears indirectly through the interaction of frame size with the rate controller.

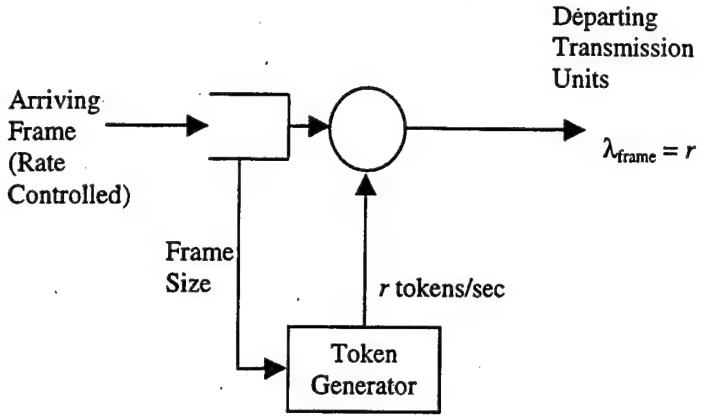

Figure V.6: Cell Level Traffic Smoothing.

Practical implementation of the smoothing scheme shown in Figure V.6 for layered video raises many additional issues. With layered video traffic, smoothing can be performed on a per-layer basis or on the entire video stream. Only the latter approach is examined here because of its simplicity. The next concern is how cells from each layer are interleaved for transmission. With FCFS scheduling, the order is unimportant but appears to play an important role in priority-based scheduling. A final concern is the identity of the transmission unit mentioned above. Smoothing must be implemented at some point prior to network entry, which in turn implies smoothing must be performed at the source node [93]. Examining the ATM protocol stack in Figure II.2, the ATM layer marks the beginning of the network since the ATM layer includes network management functionality. Therefore, ATM cells are not a suitable transmission unit unless new functionality is added to the ATM layer. Instead, smoothing must be implemented above the ATM layer, either prior to the AAL or within the AAL. In either case, a suitable candidate for the transmission unit is a higher layer PDU, either an application PDU or an AAL-PDU. However, assuming that processing times within the lower layers are fixed, a suitable scheme can be devised that provides an effect equivalent to smoothing transmission of ATM cells within the ATM layer.

2. Predictive Smoothing

The primary drawback of the smoothing scheme identified in Figure V.6 is the added transmission delay. The delay consists of two components. First, since transmission is smoothed over an entire frame interval, a delay equal to the frame interval is inserted into the transmission path. For video-on-demand applications, the added delay poses no difficulty, but for interactive applications the delay becomes of greater concern as the frame rate decreases. For example, at 10 fps, a delay of at 100 ms is created. To meet the ITU-T standard of 150 ms for interactive applications [5], total transmission and queuing delay cannot exceed 50 ms. While stringent, this delay requirement appears feasible for the LOS wireless network considered here since transmission delays are small. The second delay component is due to the need to wait for the entire frame to be encoded before the token rate r is determined.

While the buffering delay is set by the frame rate, the delay due to frame encoding may be reduced by instead predicting the size of the compressed frame [94]. The predicted value is used to set the token rate such that transmission units are transmitted immediately as they become available from the coder. Taking advantage of the correlated nature of the compressed video stream, the size of the current frame $B(n)$ can be predicted from the sizes of the last P frames:

$$B(n)=\sum_{k=1}^{P} a_k B(n-k), \tag{V.18}$$

where a_k are the filter weights. Several predictive techniques appear feasible for determining the weights in Eq. (V.18). Work by Randhawa and Hardy [95] on VBR video traffic streams found good results using the LMS algorithm. Another approach considered by the author [94] here determines the weights adaptively using the RLS algorithm [96] for the next frame during transmission of the current frame. The RLS algorithm offers the advantage of requiring less information about the input sequence than does LMS. Given that some error is present in the estimate, two scenarios are possible during each frame interval. If the prediction is high, the transmission buffer empties before the frame interval expires. However, with sufficient accuracy, the

benefits of smoothing cell delivery are still realized. If the prediction is too low, the buffer still has cells for transmission from the current frame, which are added to the size estimate for the next frame. To account for the learning period of the RLS algorithm, the predicted values are only employed once the prediction error has dropped below a threshold value related to the encoder delay.

Using simulated VTC traffic, the RLS algorithm appears reasonably accurate in predicting compressed frame sizes. To validate the approach proposed above, the RLS algorithm was used to predict frame size for a VTC sequence generated from a modified version of the minisource model proposed by Sen et al. [88] described in Section V.A. They reported that fluid source modeling with 20 minisources per video source produced reasonable agreement with queuing simulations. Using the minisource parameters reported for VTC traffic in [27], Sen's model was modified by replacing each minisource with a statistically equivalent first order autoregressive process to remove the affect of quantization. Figure V.7 shows the resulting video stream along with the predicted values using three taps (using more taps did not improve accuracy). Figure V.8 shows the corresponding prediction error. Assuming an encoder delay of 20-40 ms, predicting frame sizes in this manner saves effectively 15-35 ms of delay per frame. These results are only valid for low activity video, such as that found in VTC. Tests with video containing a large number of scene changes and/or a high degree of motion generated larger prediction errors.

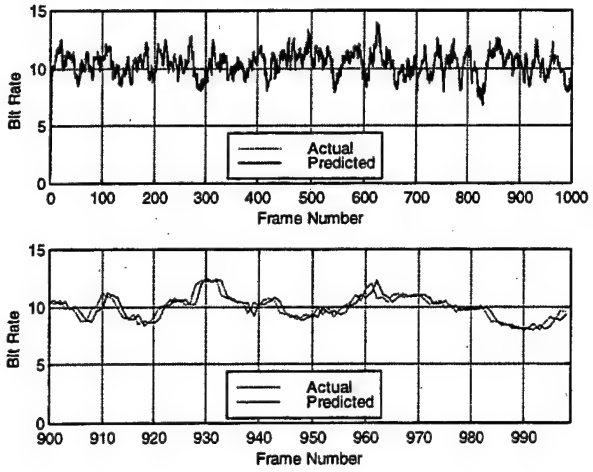

Figure V.7: Simulated and Predicted VBR VTC Sequences Using RLS.

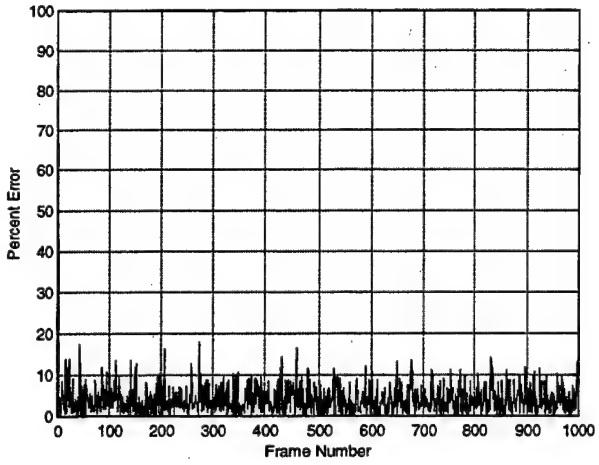

Figure V.8: Prediction Error for 3-Tap Filter Using RLS.

3. Interleaving/Transmission Order

The hierarchical nature of layered video facilitates the use of priority-based scheduling schemes within the network to ensure that service is granted in accordance with perceptual importance. Previous work by Luo and Zarki [16] indicates that when

transmitting priority-based traffic, performance is degraded by the degree to which cells from different priority classes are segmented together. Their results suggest that while smoothing the interarrival times of cells to the ATM multiplexer can increase queuing efficiency, the order in which cells are transmitted from each layer must also be considered to promote priority-based scheduling.

The need to smooth cell traffic across layers is demonstrated by a simple example. Consider a layered video stream in which each layer has a distinct priority. Further assume that all cells for each layer from a GOB or a frame are concatenated prior to transmission. Each layer can be viewed as a separate cell flow as depicted in Figure V.9. As a result, each cell flow now appears more bursty than the parent cell flow. If cell flows of similar priority from different connections become correlated in time, the result is higher cell loss in that priority class. This is analogous to the problem with correlation between bursty video streams originally addressed by Dixit [89]. If high-priority cell arrivals from different connections become correlated at the ATM multiplexer, the expected benefit derived from prioritization is denied since only one priority class is available for scheduling. Another viewpoint is that a connection is given only a finite number of service opportunities in a given time interval. Giving higher-priority cells precedence is only effective if those cells are available in the queue at the instant of scheduling. Concatenating priority levels, or in this case cells from different layers, creates time intervals wherein higher priority cells are not arriving into the queue and therefore creates periods where they are not available for service. Obviously, the impact is controlled to some extent by buffer size.

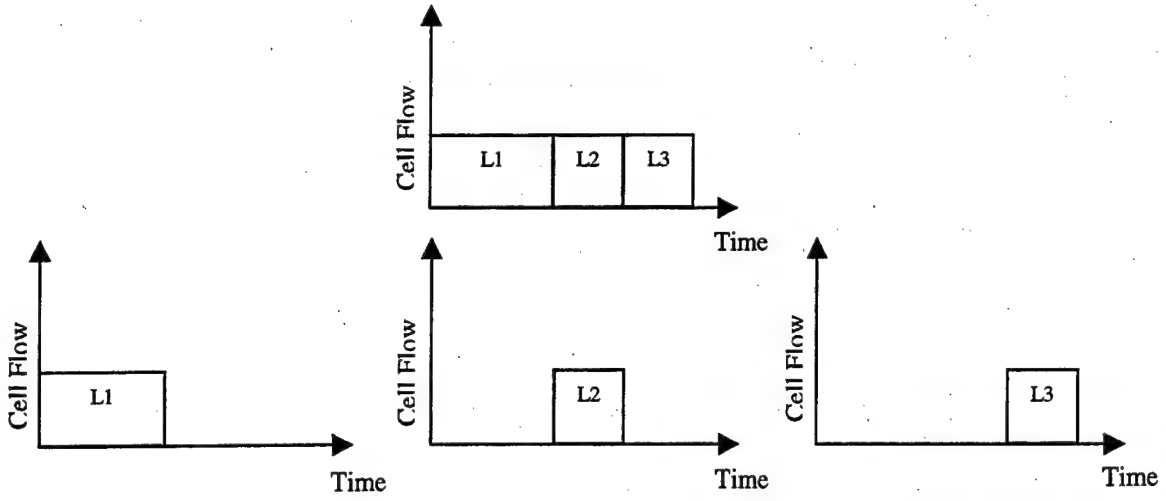

Figure V.9: Equivalent Cell Flows for Prioritized Traffic.

We propose smoothing across layers by interleaving cells from the different layers. Interleaving maximizes the average distance between cells (in time) from a particular layer and provides the maximum smoothing of cell interarrival times for that layer. However, this in no way affects the interarrival times between adjacent cells in the connection, which are set solely in response to frame size. The degree of interleaving depends on the ratio of cells available from each layer. Let us consider three layers with the average bit allocation among layers of 4:2:2. Figure V.10 presents several possible interleaving arrangements ranging from complete segmentation to maximum interleaving, where C# identifies an individual cell and # its parent layer.

...	C1	C1	C1	C1	C2	C2	C3	C3	...
...	C1	C1	C2	C3	C1	C1	C2	C3	...
...	C1	C2	C1	C3	C1	C2	C1	C3	...

Figure V.10: Several Possible Interleaving Schemes Given a 4:2:2 Ratio Among Layers.

Cell interleaving is accomplished by modifying the leaky bucket mechanism shown in Figure V.6 as follows. A separate queue is maintained for each layer. The token rate is generated based on the accumulative size of each layer (which is just the frame size). Each time a token is available, an interleaver selects a cell from one of the buffers in accordance with the desired interleaving scheme. The modified smoothing technique is shown in Figure V.11. Practically, the bit allocation among layers varies over time; therefore, creating an *a priori* fixed interleaving scheme may be impractical. A reasonable approach, given the correlation between successive frames, is to assume that the bit allocation among layers for the last frame is indicative of the allocation in the current frame. Then the ratio of bits allocated to each layer can be used to create an interleaving pattern for the current frame. A simpler low-delay approach would impose a round-robin order on transmissions for those layers having cells available for transmission.

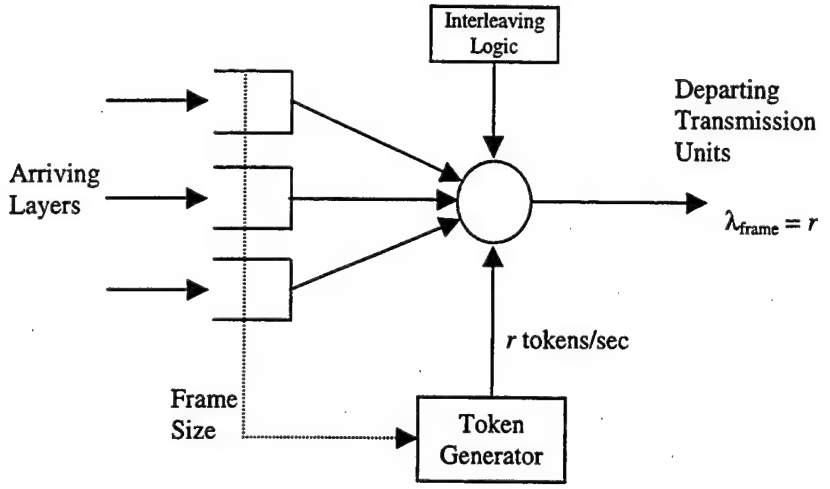

Figure V.11: Cell Smoothing and Cell Interleaving Over One Frame.

Further consideration of the impact of cell interleaving on scheduling performance is deferred until the next chapter. In Chapter VI, after proposing a new scheduling algorithm for layered video, the effect of cell interleaving is demonstrated through traffic simulations by regulating the interleaving pattern within the traffic model.

4. Smoothing: Single VCI Case

The discussion above addresses smoothing over three logical entities in the video stream: frames, layers, and cells. Smoothing frames to satisfy a bit constraint is accomplished at the application layer by a rate controller, such as the scheme described in Chapter IV. The remaining smoothing is accomplished by using either the leaky bucket smoother shown in Figure V.6 or the leaky bucket interleaver shown in Figure V.11. Placement of the smoother is not an arbitrary decision. As discussed earlier, all smoothing must be done prior to network entry. This leaves the option of implementing the leaky bucket mechanism either prior to the AAL or within the AAL. Each approach has merits, but implementation prior to the AAL negates the need for further modifications to the AAL. In either case, implementation carries the smoothing effect over to the ATM cell flow assuming fixed delays throughout the protocol stack.

Working with the AAL2 scheme proposed for multiplexing a layered video source over a single VCC (see Section II.D.4), the first approach is to insert the leaky bucket mechanism at the application layer prior to the SSCS sublayer as shown in Figure V.12. The bit stream issuing from the coder for each layer is buffered at the smoother. As tokens become available, the smoother selects a transmission unit from one of the buffers in accordance with the interleaving scheme and forwards it to the appropriate AAL SAP. As discussed in Section II.D.4, a block size of 44 octets works well with the CPCS sublayer since, with overhead, this fits within exactly one ATM cell. Accordingly, an appropriate transmission unit is 44 octets for the smoother. Within the AAL, the SSCS sublayer merely hands the PDU over to the CPCS sublayer for encapsulation. Since the smoother multiplexes the flow of PDUs into the AAL, no explicit support for multiplexing is required within the CPCS sublayer. The only other requirement is that the coder must signal the smoother at the start of each new frame, so the token rate can be recalculated.

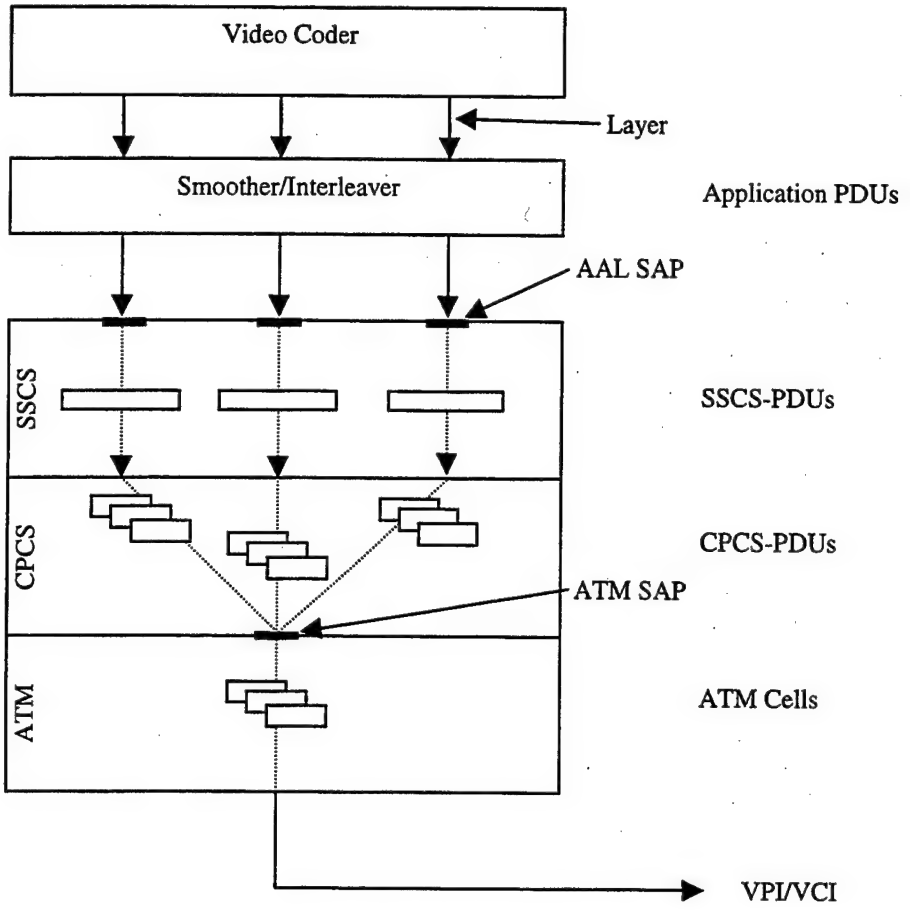

Figure V.12: Smoothing at the Application with a Single VCI.

The second approach is to insert the leaky bucket within the CPCS sublayer prior to the ATM SAP as shown in Figure V.13. Here, CPCS-PDUs are buffered individually for each layer and handed to the ATM SAP as tokens become available in accordance with the interleaving scheme. Obviously, the appropriate transmission unit in this case is the CPCS-PDU. In this case, the smoother performs multiplexing within the CPCS sublayer. Once again, the video coder must signal the smoother after each frame to allow computation of a new token rate.

Independent of whether smoothing is implemented prior to the AAL or within the SSCS sublayer, *arbitrary* cell interleaving precludes segmenting cells as shown in Figure II.13 to allow network nodes the option of identifying GOB boundaries. Arguably, since a GOB is some fraction of a frame, one-ninth of a frame in the coder presented in Chapter

IV, organizing cells into layer GOBs does provide some benefit of cell interleaving. If GOB boundaries are to be respected, the traffic smoother shown in Figure V.11 must be modified to buffer individual GOBs from each layer instead of a complete layer from each frame as originally proposed. Remaining GOBs are held at the application level until required. The interleaver services each buffer sequentially starting with the lowest layer. As each token arrives, the interleaver draws an appropriate transmission unit from the buffer until the buffer is exhausted. After all buffers are exhausted in similar fashion, all buffers are refilled.

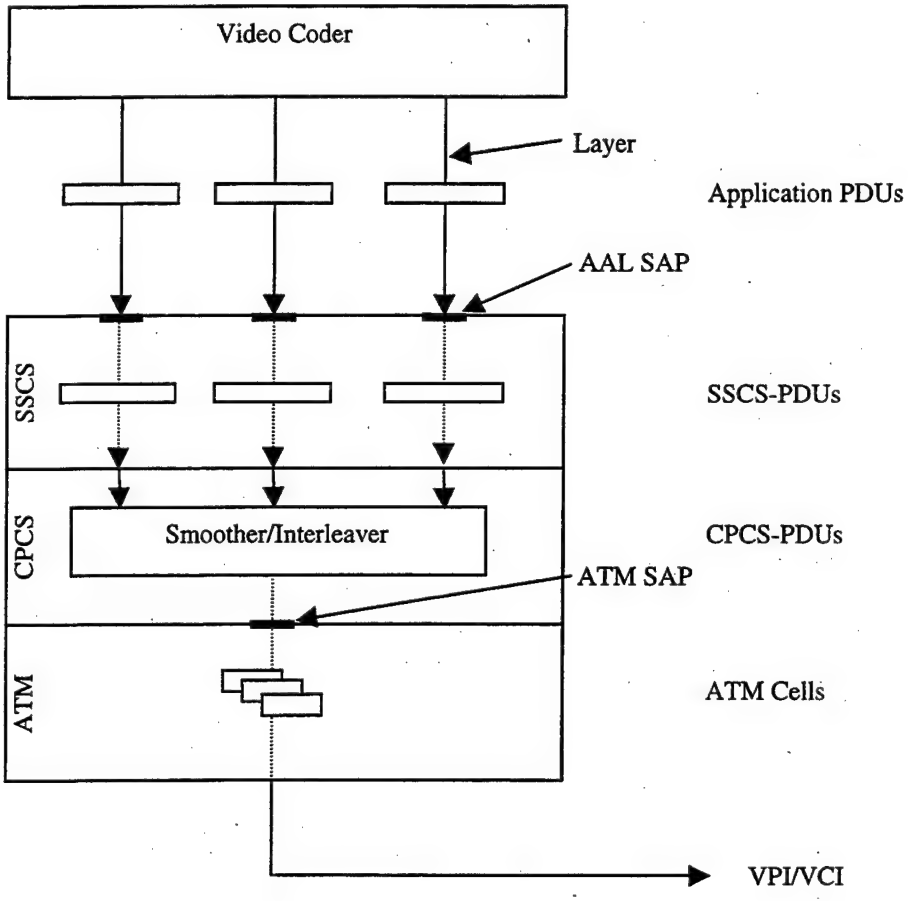

Figure V.13: Smoothing Within the AAL with a Single VCI.

5. Smoothing: Multiple VCI Case

The multiple VCI case was covered in Section II.D.3 and differs from the previous case in that a separate VCI is used to transmit each layer, and AAL5 is

employed due to the lower overhead. Implementing smoothing again involves the same considerations presented in the last section except that buffering at the application level forces reconsideration of AAL5 for the AAL protocol.

The first approach is to insert the leaky bucket mechanism at the application layer prior to the CS sublayer as shown in Figure V.14. The bit stream emrging from the coder for each layer is buffered at the smoother. As tokens become available, the smoother selects a transmission unit from one of the buffers in accordance with the interleaving scheme and forwards it to the appropriate AAL SAP. Here, using AAL5 reveals a distinct lack of efficiency. To realize the benefits of smoothing at the ATM level, processing between the smoother and the ATM layer should be minimized. This is most simply accomplished by transmitting only that amount of data that will eventually fit within a single ATM cell. If AAL5 is used, this would limit the transmission unit to 40 octets. In light of this, AAL2 offers a more efficient path since the size of the transmission unit can be increased to 44 octets. Therefore, the multiple VCI case assumes AAL2 as shown in Figure V.12. Within the AAL, further operation works as described in the single-VCI counterpart except that the CPCS-PDUs are not multiplexed within the CPCS sublayer and are instead transmitted to their respective ATM SAPs.

The second approach is to insert the leaky bucket within the SAR sublayer prior to the ATM SAPs as shown in Figure V.15. Here, SAR-PDUs are buffered individually for each layer and handed to the ATM SAP as tokens become available in accordance with the interleaving scheme. Obviously, the appropriate transmission unit in this case is the SAR-PDU. Otherwise, operation proceeds as described for the single-VCI case except that released SAR-PDUs are assigned to the ATM SAP appropriate for that layer.

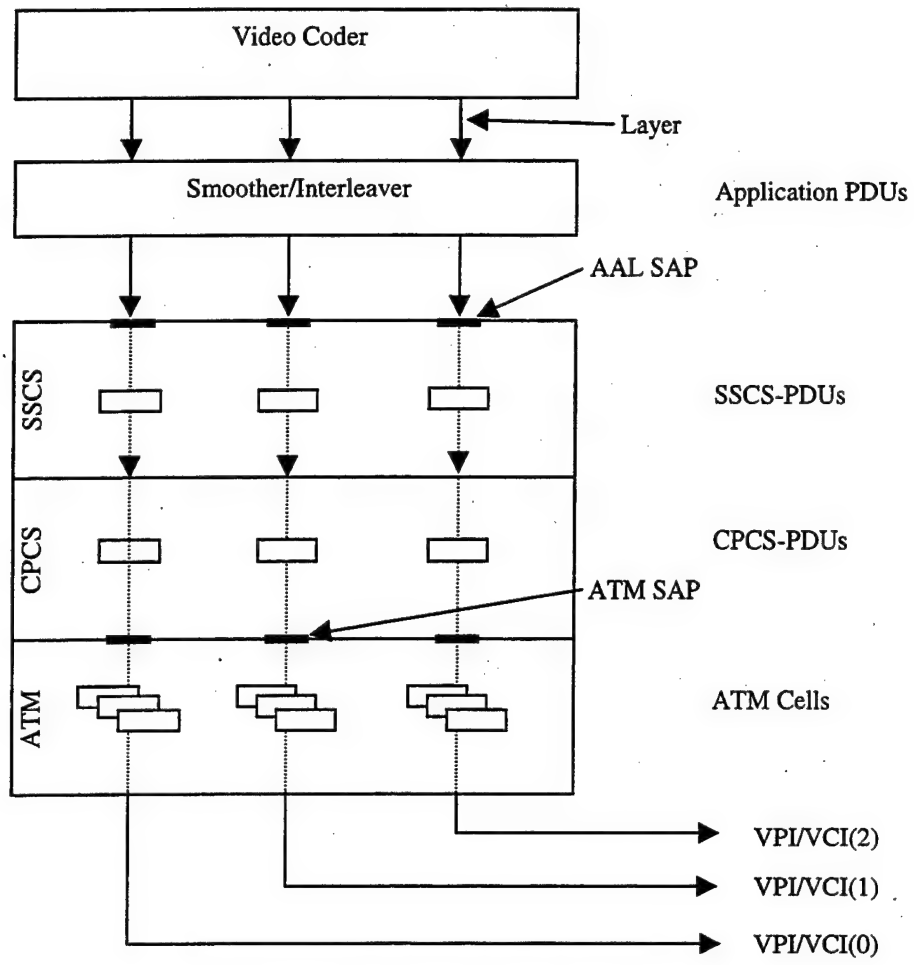

Figure V.14: Smoothing at the Application with Multiple VCIs.

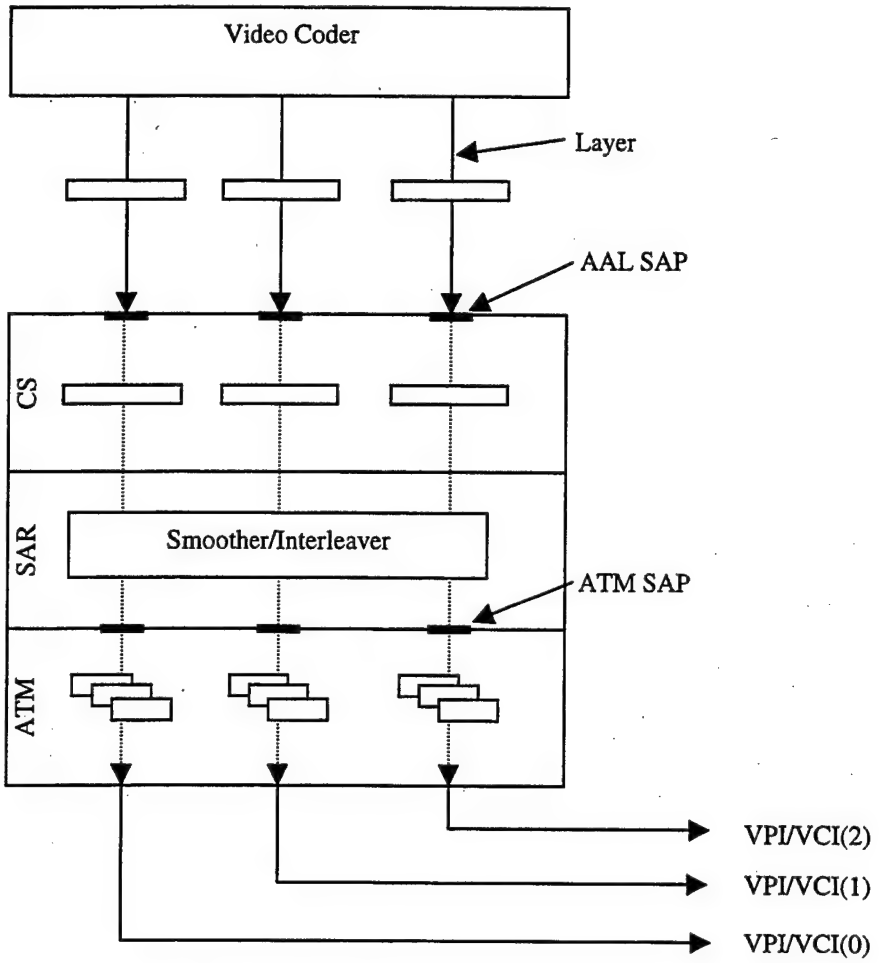

Figure V.15: Smoothing Within the AAL with Multiple VCIs.

6. Smoothing Results

The bandwidth conservation produced by this smoothing may be evaluated by constructing a histogram model for the controlled and uncontrolled sequences shown in Figure IV.26. The estimated CLR for each stream is shown in Figure V.16 using a buffer size of 10 cells, which guarantees that queuing delay does not exceed 50 ms (the significance of this value is discussed below). CLRs were calculated using Shroff's [15] ad hoc analysis technique. The result demonstrates that the rate-controlled stream requires far less bandwidth to guarantee a given QoS although the difference would lessen to some extent when mutliplexing multiple video streams. Results for three multiplexed homogenous sources are shown in Figure V.17 and reveal an even more

dramatic gulf between the rate-controlled and uncontrolled sources. While the histogram approach does incorporate frame-by-frame smoothing, the difference in queuing performance demonstrated here is attributable to the affect of the rate controller on the video stream's histogram.

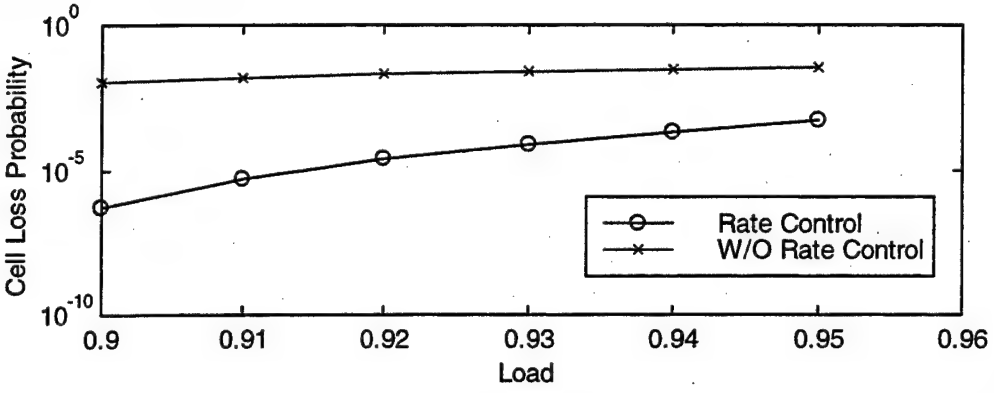

Figure V.16: Estimated CLR for Rate-controlled and Non-rate-controlled VTC Traffic.

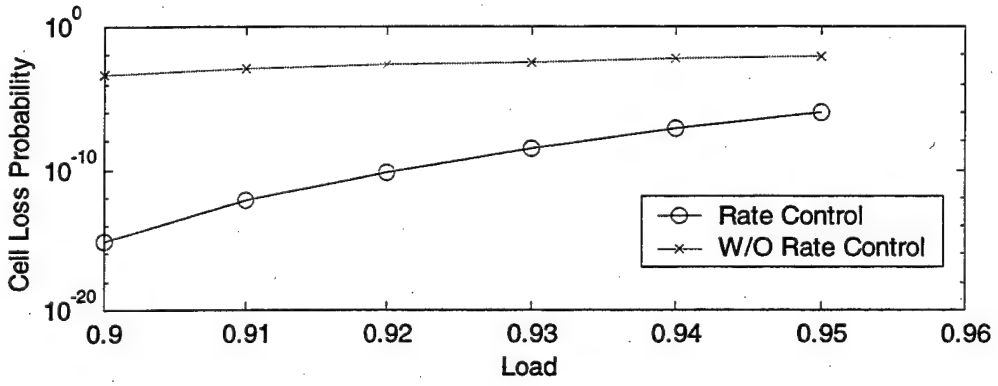

Figure V.17: Estimated CLR for Rate-controlled and Non-rate-controlled VTC Traffic (3 sources).

To illustrate the affect of smoothing at the cell interarrival level, an OPNET simulation involving three homogenous sources was created. Two cases were considered: individual frames were smoothed deterministically, that is cell interarrivals

were of fixed duration for each frame, and individual frames were smoothed in a Poisson fashion. In the latter case, cell interarrivals were distributed exponentially such that the average arrival rate equaled the size of the frame. While smoothing in Poisson fashion does not capture the higher bursty behavior described in Dixit's work [89], smoothing in this fashion does reveal the affect of randomness of queuing efficiency. The results for several different traffic loads is shown in Figure V.18. In each case, deterministic smoothing yields better performance for a given load.

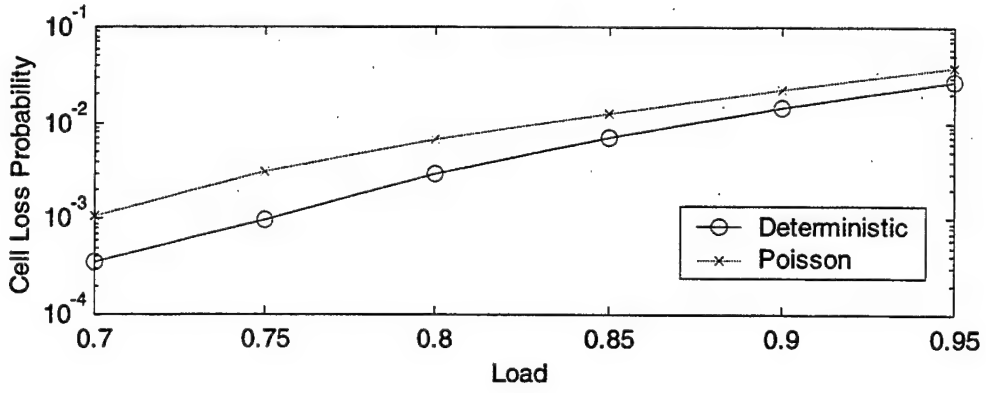

Figure V.18: Deterministic and Poisson Intraframe Smoothing.

The final issue examined is the affect of cell interleaving or layer smoothing on scheduler efficiency. Again three homogenous sources were considered, but the multiplexer implemented a service discipline using the layered scheduling algorithm discussed in the next chapter. For a given traffic load of 0.8 and a bit ratio of 2:1:1 among the three layers, queuing performance was examined for different levels of concatenation in the base layer. Specifically, run lengths of 2, 4, 6, and 8 were examined. The results are shown in Figure V.19. As the run length of cells from the highest priority layer are increased, the CLR is observed to rise. At the same time, the CLR from the lower priority layers decreases until the largest run length. The results indicate that minimizing the run length of the higher priority cells gives better performance as anticipated. Based on observations of the queuing behavior, the decrease in CLR results

from the scheduler having easier access to higher priority cell, which maximizes scheduling opportunities for those cells.

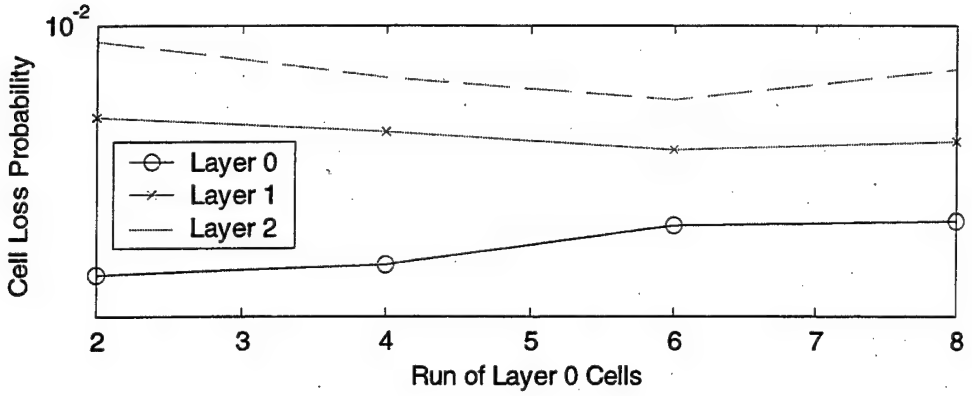

Figure V.19: Effect of Cell Concatenation of Scheduler Performance.

This chapter examined the use of smoothing to increase multiplexer efficiency and thereby lower bandwidth requirements for multiplexed VBR traffic. Smoothing was considered at the three time scales: frames, layers, and cells. Rate control effectively smoothes at the frame level and is a part of the transmitting application. Smoothing across layers and cells requires insertion of a smoothing mechanism in the transmission path prior to network entry. A smoothing mechanism based on the leaky bucket algorithm was presented, and its implementation for layered video traffic was explored. A video traffic model for smoothed video traffic was also presented and is used in simulations presented in the next chapter.

VI. SCHEDULING LAYERED VIDEO TRAFFIC

The final element in delivering layered video is designing a network scheduler that exploits the perceptual hierarchy inherent in layered video by prioritizing delivery to mitigate the affects of congestion. As shown in Figure VI.1, a network scheduler is implemented at each switch within the network and controls access to a network resource, namely the capacity of the outgoing line. The switch's scheduling algorithm is responsible for sharing the line capacity amongst several customers, a difficult problem if each customer has different QoS requirements. The manner in which the scheduler is implemented determines the maximum number of customers that can be served within the available capacity. Therefore a tight relationship exists between the scheduling policy and the admissions policy.

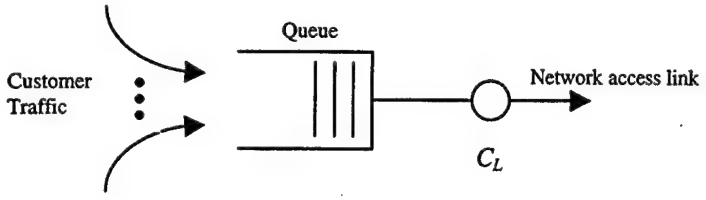

Figure VI.1: A Switch Controlling Access to a Network.

A scheduling policy consists of a queuing discipline and optionally a cell discard policy. Examining the queue in Figure VI.1, a scheduling policy determines how to queue cells awaiting service and the order in which to serve cells. The choice of scheduling policy directly impacts the ability to meet QoS in two ways. First, queued cells awaiting service experience delay due to the gap between their arrival time and service time. Second, queues are finite, and if the node experiences a cell burst, the queue may fill causing all further cell arrivals to be discarded. A larger queue has a smaller probability of experiencing cell loss but imposes potentially greater delays on arriving cells. The scheduling policy may be coupled with a cell discard policy, which helps guarantee QoS and responds to congestion. One example of a cell discard policy is

to monitor queue length, and if the length exceeds a threshold, lower priority cells are discarded to prevent congestion.

This chapter provides a scheduling mechanism that incorporates three criteria described above; a scheduler that guarantees QoS, performs optimal scheduling for different traffic classes, and prioritizes cell delivery from each layer as required. The discussion starts with a short survey of scheduling algorithms that incorporate QoS requirements into scheduling decisions. The discussion leads to the STEBR algorithm proposed by Uziel [39]. STEBR is an optimal scheduling algorithm for heterogeneous traffic and is used as the basis for designing a scheduler for layered traffic. STEBR is modified for layered traffic by incorporating the notion of priority *within* a connection. Prioritization is brought in through a filtering mechanism that subordinates the QoS granted to lower priority layers to that received by higher priority layers. A partial GOB discard scheme is presented that drops unusable cells for increased effective bandwidth utilization. Finally, OPNET simulations are presented to verify the validity of the proposed schemes.

A. SCHEDULING CRITERIA

The specific problem examined here is to determine a scheduler design for a layered video traffic stream that meets several criteria. The simplest criterion is that the scheduler should meet the QoS requirements for the video stream. Although the layered coder is designed for robustness, limits on the cell loss rate help deliver a less distracting viewing experience by limiting fluctuations in reconstructed quality. Since VTC is an interactive application, limits on scheduling delay are also required. Since QoS guarantees are desired, a scheduling policy, such as first come, first serve (FCFS), is clearly impractical. With FCFS service, arriving cells are handled merely by servicing the cell at the head of queue. The requirement here is a scheduling policy that integrates each connection's QoS requirements into scheduling decisions.

Given a low-bit-rate networking environment, another criterion is that the scheduling policy maximize utilization of resources. Maximum utilization may be

viewed as maximizing throughput or, alternatively, supporting the largest number of customers, each with their own unique QoS constraints. Given two scheduling policies, the more optimal policy is the one that admits the largest number of connections. To demonstrate the effect of scheduling policy, consider a network with two services classes, each representing a fixed set of QoS parameters. The admissible region is two-dimensional, representing all allowable combinations of connections from each service class. As described above, a FCFS scheduling policy does not inherently take QoS into account and, therefore, gives the smallest admissible region. An optimal scheduling policy gives the largest admissible region. This situation is shown in Figure VI.2 for the two service classes. As the number of connections for either service class goes to zero, both scheduling strategies give the same performance. In general, if k service classes are defined, the resulting admissible region is k-dimensional. Finding and verifying an optimal scheduling strategy is difficult in this case.

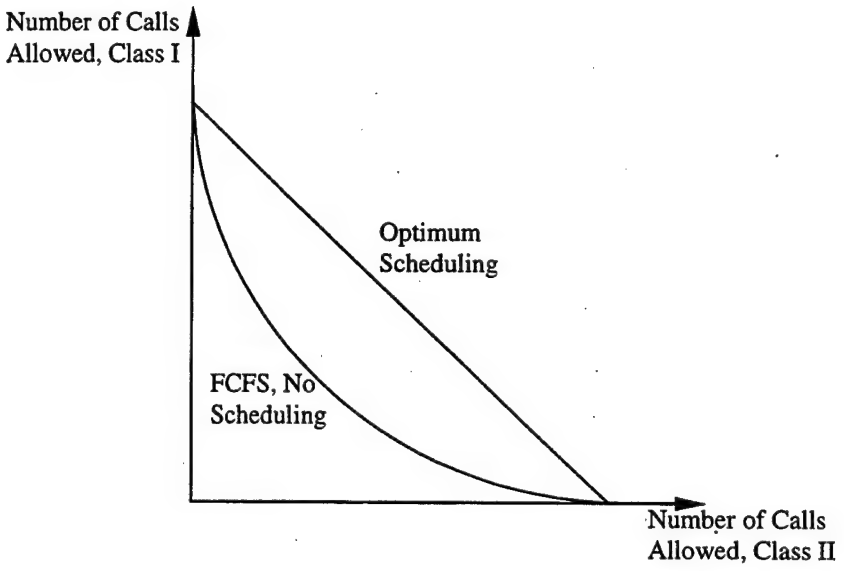

Figure VI.2: Admission Regions for FCFS and an Optimal Scheduling Policy.

The final criterion required of the scheduler is to exploit the hierarchical nature of the layered video stream in choosing which cells to service and which to deny service.

This goes beyond the notion of guaranteeing QoS to a connection. Any time the scheduler has insufficient bandwidth to transmit all waiting cells, perhaps due to a traffic burst or congestion, cell loss is inevitable. The problem is devising an intelligent mechanism for deciding which cells in a connection to service, or alternatively, which cells to deny service. With a layered video stream, the relative perceptual importance of each layer imposes an inherent transmission priority. Any loss in the base layer is catastrophic: some portion of the picture cannot be reconstructed. Losses from the enhancement layers only degrade reconstructed quality. Therefore, an intelligent service policy is to favor transmitting cells from the higher priority layers at the expense of those from lower priority layers as required. This points to a hierarchical service policy in which the layered video *connection* is assigned a certain QoS, i.e., a certain amount of bandwidth is allocated for all layers to share. However, during periods when the QoS cannot be maintained, a transmission priority is enforced that allocates bandwidth to the more perceptually important layers. In this manner, congestion causes the reconstructed video to degrade gracefully but remain viewable.

Another facet of the layered stream to consider is the hierarchy placed on the organization of the bit stream within each layer. In particular, a decoder can only resynchronize after cell loss at select points within the bit stream. Therefore, a single cell loss may render subsequent cells unusable to the decoder. Since these cells could otherwise hinder transmission of other viable cells, a suitable cell discard scheme that discards unusable cells regardless of QoS constraints could increase effective utilization of the outgoing link.

B. QOS SCHEDULING ALGORITHMS

Referring to Figure VI.3, scheduling algorithms allocate bandwidth amongst different traffic sources according to the QoS required by each source. As indicated above, the FCFS policy is the simplest scheduling policy but treats all service classes equally and is not suitable for an integrated services network. This section describes several classes of scheduling algorithms, starting with an overview of early efforts and

ending with two novel methods proposed by Uziel [39]. Uziel also provides a more comprehensive review of scheduling algorithms.

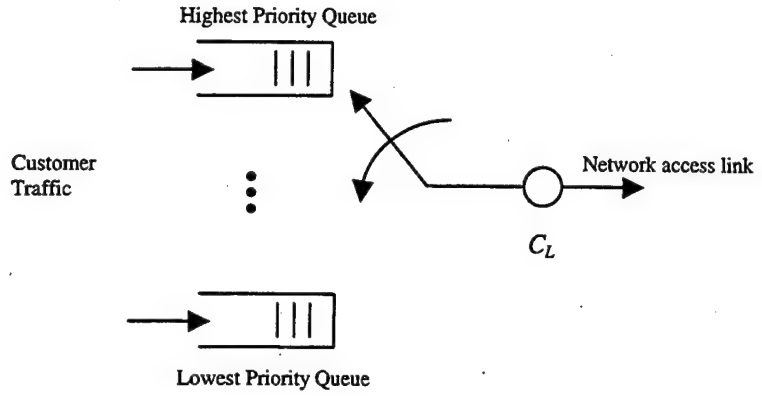

Figure VI.3: Scheduling Different Priority or QoS Classes.

1. Reservation Schemes

Static-priority-scheme (SPS) algorithms differentiate between differing QoS requirements through priority assignments. Each traffic source is assigned a priority, and each cell is tagged with the appropriate priority. Cells are served in priority order while cells with the same priority are served using a FCFS policy. This policy may be envisioned by replacing the single queue in

with a queue for each priority level as shown in Figure VI.3. Higher priority queues are served until emptied, and cells within each queue are served FCFS. SPS algorithms are simple to implement and provide flexibility in serving different traffic classes but provide poor performance in certain situations. For example, if high priority cells have higher delay (maxCTD) requirements while lower priority cells have stricter maxCTD, the low priority cells will receive poor service and face potentially high loss rates.

A related approach is bandwidth reservation in which traffic sources are guaranteed a bandwidth allocation in proportion to a traffic statistic or QoS requirement. Bandwidth may be allocated among n traffic sources by simply dividing the capacity evenly, apportioning bandwidth according to mean bit rate or by a weighted combination

of the mean and variance of the bit rate [97]. However, each of these approaches fails to account for QoS requirements and offers only marginal performance over FCFS. A better approach is to assign bandwidth as a function of required QoS [98]. For example, if each of the traffic sources has a maxCTD of T_i, then each source receives a guaranteed bandwidth of

$$BW_i = \left(T_i \Big/ \sum_{j=1}^{n} T_j \right) C_L, \qquad \text{(VI.1)}$$

where n customers compete for service, and C_L is the line capacity. The drawback to static allocation and bandwidth reservation is that both may leave the server underutilized with bursty traffic since spare capacity cannot be reallocated among sources.

2. STE and BCLPR

The shortest time to extinction (STE) algorithm proposed by Panwar et al. [99] handles traffic sources with deadlines (maxCTDs). The goal is to maximize the fraction of cells entering service prior to their respective deadlines or, equivalently, to minimize cell loss due to expiration for *G/D/*1 queues. Each cell entering the queue is assigned a deadline or time of expiration (ToE) that diminishes the longer the cell waits in the queue. Service periods are divided into slots; each slot represents the time required to service one cell. At the beginning of every service slot, the ToE is updated for each cell. Cells missing their deadline to start service, indicated by a ToE less than the service slot, are dropped from the queue. Of the remaining cells, the cell with the lowest ToE is serviced. STE is optimal with respect to cell loss rate and is simple to implement. However, STE is not optimal for heterogeneous traffic streams, where each stream may have different maximum CLRs.

Uziel has proposed a new scheduling algorithm, the balanced-CLP-ratio (BCLPR) algorithm, that improves upon STE for heterogeneous traffic [39]. BCLPR makes scheduling decisions based on each connection's CLR requirement along with STE's approach of dropping cells that are unable to meet their service deadline. BCLPR calculates two statistics for each connection: an instantaneous CLP (ICLP)

$$ICLP[i] = \frac{DS[i]}{A[i]} = \frac{\text{Cells discarded from connection i}}{\text{Total cells arrived from connection i}}, \qquad \text{(VI.2)}$$

and a cell-loss probability ratio (CLPR)

$$CLPR[i] = \frac{ICLP[i]}{ACLP[i]}, \qquad \text{(VI.3)}$$

which compares the instantaneous CLP with the allowable CLP (the CLR quality of service (QoS) constraint). The algorithm employs the following steps at the beginning of each service interval. The ToE of each cell is calculated, and expired cells are dropped from the queue. The ICLP and CLPR statistics for each active connection are updated, and the first cell in the queue from the connection with largest CLPR is selected for service. If two or more connections have the same CLPR, a cell is selected at random from one of the connections.

Over time, BCLPR ensures that each connection is granted at least its guaranteed QoS. If a connection's CLPR exceeds one, the connection is getting less than its guaranteed QoS, and the connection has a greater chance of receiving a service slot from the scheduler. A connection with a CLPR less than one is getting better QoS than guaranteed, so it will receive correspondingly less service from the scheduler. Over time, the average CLPR for each source approaches the same value. The proximity of the value to one depends on the scheduler loading.

Summarizing, STE is optimal with respect to cell loss rate when considering homogeneous traffic. BCLPR employs cell loss rates in scheduling decision, which improves performance with heterogeneous traffic. However, BCLPR does not employ the proximity of a cell to expiration in scheduling decisions. This leads to poor performance for bursty traffic wherein the scheduler may choose to service a connection ignoring a burst of cells from another connection nearing expiration in the queue. In fact an oscillation can arise such that a connection only receives service following the loss of a cell burst, which degrades system throughput [93].

3. STEBR

The STE with BCLPR (STEBR) scheme proposed by Uziel [39] corrects this behavior by considering both the instantaneous loss rates experienced by each source, the deadlines of cells within the queues, and the expected losses if service is denied given that there are no further arrivals. STEBR makes optimal scheduling decisions in the sense that no other algorithm for a single-queue single server system has a larger admissible region. STEBR employs a predictive cost function associated with each connection's current CLPR that increases with the number of cells discarded.

Each cell in the queue is assigned a cost representing the future impact of service denial on the parent connection's CLPR. The cost for connection i is represented as:

$$Cost[i] = CLPR[i]\Big|_{\substack{\text{if an additional}\\ \text{cell is discarded}}} = \frac{(DS[i]+1)}{A[i]\times ACLP[i]} = CLPR[i] + (A[i]\times ACLP[i])^{-1}, \quad \text{(VI.4)}$$

where $DS[i]$ is the number of cells discarded from connection i, $A[i]$ is the number of arrivals from connection i, $ACLP[i]$ is the desirable CLP for connection i, and $CLPR[i]$ indicates how well the connection is being serviced. The cell closest to expiration is assigned this cost. Working towards the back of the queue, newer cells are assigned an incrementally greater cost in a linear fashion, where the increment is:

$$\Delta_i = (A[i]\times ACLP[i])^{-1}. \quad \text{(VI.5)}$$

Since any scheduling decision made for the current service slot may lead to expiration of other cells expiring due to denial of service, STEBR schedules the cell that minimizes the overall system cost for all connections using the above cost function.

At the start of each service slot, the queue is scanned, and cells that have expired are dropped. Next, the CLPR for each connection is updated, and each cell is assigned a cost using using Eq. (VI.4) and Eq. (VI.5). STEBR then partitions the waiting cells by assigning each cell to the most future service slot in which the cell could receive service and still avoid expiration (a value of 1 indicates that the cell will expire if not granted service). The service slot for the jth cell is calculated as

$$T_k = \left\lfloor \frac{ToE[j]}{1/C_L} \right\rfloor = \lfloor ToE[j]\times C_L \rfloor, \quad \text{(VI.6)}$$

where C_L is the channel capacity in cells/second. Each service slot may have multiple cells from one or more connections. Working from the most future service slot to the current time slot, the algorithm assigns exactly one cell to each service slot by examining the cost of each cell. If a connection has multiple cells in the service slot, only the one with the maximum cost is considered. The service slot is awarded to the connection with the highest cost cell. Cells that are not selected for service are moved to the next service slot, and the procedure is repeated. This action recognizes that while a cell not selected for service in slot k will expire if deferred to a later slot, service in slots 1 to k - 1 is still feasible. The process is repeated until slot 1 is reached, which represents the current service slot. The cell awarded service in slot 1 is actually transmitted. Note that any cells originally assigned to slot 1 not selected for transmission will be discarded during the next service interval. Uziel [39] provides several examples that illustrate this process.

C. LAYERED SCHEDULING

1. QoS Filtering and the STEBR Algorithm

The strategy for a layered video connection is to have the switch first maintain the contracted QoS for the connection overall and then maintain a specified QoS for each layer within the connection. Emphasizing the connection's QoS ensures that the connection receives fair access to the bandwidth originally granted by the network. The QoS received by each layer within the connection is subordinate to the desire to preserve QoS for the higher priority layers during periods of congestion. That is, we choose to reallocate bandwidth dynamically within a connection to maintain QoS for the higher priority layers. Two schemes are examined here to reallocate bandwidth dynamically. The first scheme is to selectively employ prioritization in a hierarchical fashion: cells from lower priority layers are denied service only when higher priority layers are not receiving their guaranteed QoS and have cells awaiting service. Selective prioritization allows explicit QoS guarantees for each layer that are then relaxed for the lower priority layers during periods of congestion. As shown later, this approach offers efficient utilization of the outgoing link. However, the hierarchical dependence within the video

stream makes the transmission of base layer cells imperative. Since the lower priority layers scale quality using information in the base layer, a loss in the base layer makes related information in the lower priority layers unusable. This point is covered in more detail in the next section. In recognition of this dependence, a second scheme only transmits cells from low priority layers whenever no cells from high priority layers are awaiting service. This scheme maximizes throughput of the base layer but ignores QoS guarantees for individual layers. Dropping the low priority layers also forfeits any opportunity to employ these layers in error concealment schemes.

Selective prioritization is accomplished by an extending the STEBR algorithm discussed in the last section. The linear STEBR algorithm was chosen since it makes optimal scheduling decisions with respect to heterogeneous CBR and VBR traffic, and it imposes no service penalty on homogeneous traffic. Implementing the linear STEBR algorithm is also computationally efficient since complexity scales linearly with queue length. The extension described here is posed only with respect to scheduling of homogenous, layered traffic although the algorithm readily extends to servicing heterogeneous non-layered traffic.

Recall that the STEBR algorithm uses the CLPR as a cost function for granting service. In particular, each cell is tagged with a speculative cost that represents the increase in CLPR if service is denied to that cell. Then, the algorithm divides the queue into service intervals and schedules cells from the back of the queue forward. The cell with the greatest cost is assigned the current scheduling slot; all other cells move down to compete for the next scheduling slot until the first slot is reached. The cell winning the first slot is actually granted service.

To modify the STEBR algorithm for layered traffic, the *CLPR* for individual layers as well as their parent connections must be maintained. As each cell arrives to the queue, its connection j and layer number k are determined, and the appropriate arrival counts are updated for that connection $A[j]$ and that layer $AL[j,k]$. At the beginning of each scheduling interval, the queue is sorted in terms of decreasing ToE. Starting from the head of the queue, each expired cell is dropped if its ToE is less than the service time.

If a cell is denied service, the dropped cell counts are updated for the affected connection $DS[j]$ and the layer $DSL[j,k]$. After the queue is scanned, the current *CLPR* for each connection and each individual layer is updated. Each connection's *CLPR* is calculated using Eq. (VI.3), and the *CLPR* for each layer is calculated in an analogous manner using

$$CLPRL[j,k] = \frac{DS[j,k]}{A[j,k]ACLP[j]}, \tag{VI.7}$$

where $ACLP[j]$ is the allowable cell loss for connection j. Equation (VI.7) assumes that each layer is assigned the same QoS as the connection. However, Eq. (VI.7) could easily be modified to apply a different QoS to each layer.

The STEBR algorithm is incorporated into layered scheduling in the following manner. At the beginning of each scheduling interval, STEBR is applied to determine which connection receives service based on the QoS granted to the connection so far and the associated cost function. Layering does not explicitly play a role in determining the connection that receives service. After a connection is granted access to the current time slot, the next decision is to determine the layer within the connection that receives service. The procedure is to compete for service based on each layer's current *CLPR*. First, since a layer without cells present in the queue does not need to compete for service, the *CLPR*s for these layers in the winning connection are zeroed out. Second, remaining layers with non-zero *CLPR*s are filtered to prioritize transmission consistent with the perceptual importance of each layer. The two schemes mentioned above are implemented using the filtering algorithms shown in Figure VI.4. With bandwidth sharing, the intent is to give a higher priority to the more perceptually important layers only when those layers are not receiving their desired QoS. Otherwise, all layers are treated in an equal manner. This QoS-based prioritization is implemented by zeroing out the CLPR of lower priority layers whenever a higher priority layer is not receiving the requisite QoS as indicated by a CLPR of greater than one. With priority sharing, a lower priority cell only receives service if no higher priority cells are available for service within the queue. This is accomplished by zeroing out the CLPR of lower priority layers whenever the CLPR of higher priority cells is non-zero, which indicates that those layers

have cells available for service. The filtering algorithms shown in Figure VI.4 assume that each connection has three layers.

```
if (CLPRL[j,0] > 1){
        CostL[j,1] = 0;
        CostL[j,2] = 0;
}
else if (CLPRL[j,1] > 1){
        CostL[j,2] = 0;
}
```

a.

```
if (CLPRL[j,0] > 0){
        CostL[j,1] = 0;
        CostL[j,2] = 0;
}
else if (CLPRL[j,1] > 0){
        CostL[j,2] = 0;
}
```

a.

Figure VI.4: Cost Filtering per Layer for a) Bandwidth Sharing and b) Priority Sharing.

After filtering, the slot is assigned to the cell from the layer with the highest CLPR. At this point, two options were explored. Earlier work with the BCLPR algorithm indicated that selecting cells deep within a queue has a deleterious effect on throughput [93]. Selecting cells without regard to queue position may lead to the situation in which cells on the verge of expiration are ignored to service a cell from a connection with a higher cost even though that cell is in no immediate danger of expiration. The STEBR algorithm [39] corrects this by comparing the cost of denial of service for each connection on a global basis. However, the filtering algorithms re-introduce this problem to a certain extent by bypassing cells from a lower priority layer to service cells from higher priority layers as needed. Arguably, this is intentional since without the higher priority layers the lower priority layers produce no benefit to the receiver, and a lower throughput is acceptable to ensure that the appropriate cells are delivered. The tradeoff between throughput and priority service is examined by implementing service deferral. The ToE of the cell selected for service during the current time slot is examined. If the ToE indicates that the cell is not due to expire during the next time slot, service is deferred and the cell closest to expiration from that connection is selected for service instead. Service deferral therefore reverts back to STE [99] within a connection whenever possible.

A complete summary of the algorithm is given in Figure VI.5. An OPNET model that implements STEBR for layered video traffic is given in Appendix A.

1. Sort the queue in order of increasing ToE from the head of the queue.
2. Scan the queue from head to tail. For each cell:
    a. Calculate the cell's ToE.
    b. If the ToE is less than the service interval:
        i. Discard the cell.
        ii. Increment $DS[j]$ and $DSL[j]$.
3. Update *CLPR*, *CLPRL*, and Δ for each connection and layer using Eq. (VI.1) through (VI.4).
4. Assign a connection cost to each cell using Eq. (VI.5) and Eq. (VI.6).
5. Assign each cell to a tentative time slot $n = \lfloor ToE \times C_W \rfloor$.
6. Assume that after step 5, N time slots are allocated.
7. For each time slot n from N down to 1:
    a. For every cell i in that time slot from connection j, layer k:
        i. If $Cost[j] \geq 0$:
            1. Increment Extra_Cells[j].
        ii. Else:
            1. Set $Cost[j]$ = Cell_Cost[i].
    b. Find the largest $Cost[i]$. Assume the connection is j_x.
    c. If $Cost[i] > 0$, there is at least one cell awaiting service.
        i. If Extra_Cells[j_x] = 0:
            1. Set $Cost[j_x] = -1$.
        ii. Else:
            1. Decrement Extra_Cells[j_x].
            2. Reduce $Cost[j_x]$ by Δ_j.
8. Connection j_x is assigned the time slot.
    a. For each layer k with no cells enqueued, set $CLPRL[j_x,k] = 0$.
    b. Filter the cost for each layer using Figure VI.4.
    c. Assume winning layer is k_x:
        i. With service deferral:
            1. If $\lfloor ToE \times C_W \rfloor > 2$ for the selected cell:
                a. Service the cell from j_x with the lowest ToE.
            2. Else:
                a. Service the first cell from layer k.
        ii. Otherwise service the first cell from layer k.

Figure VI.5: Modified STEBR Algorithm.

2. GOB Dropping

As discussed in Chapter III, correct decoding of the compressed video bit stream requires that the decoder stay in sync with the bit stream. Bit errors and dropped cells interrupt the decoding process and force the decoder to scan the bit stream until a distinctive codeword is found to reset the decoding process. This is part of the rationale for imposing a logical hierarchy on the bit stream (see Figure III.1). For the coder proposed here, the information required to start the decoding process includes the start of the next macroblock, the scene type, and the current quantizer setting. Other coders might require additional or different information[14]. Since repeating this information consumes bandwidth, a tradeoff is forced between minimizing this overhead and the distance, in macroblocks, between resynchronization points. Most coders, therefore, support resynchronization at the start of each GOB[15]. The result is that, after a stream error, the decoder parses through the bit stream until a GOB header is recognized and restarts decoding at that point. The intervening data between the stream error and the GOB header is discarded, and the effect on the display is left up to the decoder.

The effect of dropped cells on the decoder has strong implications for the layered scheduling algorithm proposed in the last section. As shown in Figure VI.6, a cell dropped from within a GOB corrupts the GOB. Any cells remaining in the GOB are unusable since their information payload will ultimately be discarded at the decoder. In this case, making scheduling decisions based on CLPR is suboptimal since CLPR no longer represents a valid indication of the impact of denying service on reconstructed visual quality at the recipient. Indeed, dropping the remaining cells in the corrupt GOB does not further degrade the quality of the reconstructed frame beyond that imposed by the original cell drop. However, the effect of dropping the unusable cells is not merely neutral. Removing these cells from contention increases the number of scheduling opportunities to cells that still have the potential to be successfully decoded. Therefore,

---

[14] A MPEG decoder would need the frame type (I, P, or B) for example [6].

[15] H.263 has a low bit rate mode that eschews GOB headers and resynchronizes only at frame headers [56].

in a global sense, the overall effect on the reconstructed quality of all competing connections is positive especially if the released scheduling opportunities are biased toward the higher priority layers in each connection. Since the layered STEBR algorithm filters costs by layer, this is the expected outcome.

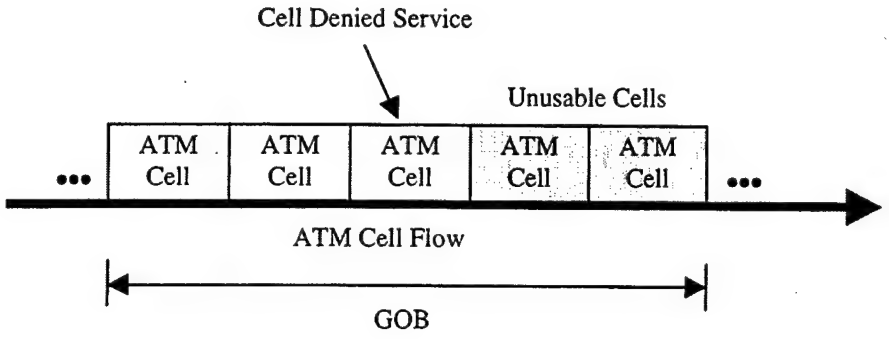

Figure VI.6: The Effect of Cell Discard on a GOB.

To illustrate these points, consider the example shown in Figure VI.7. A scheduling slot *k* contains a layer 2 cell from connection *A* and layer 0 cells from connections *B* and *C*, respectively, with the connection costs shown. The layered STEBR mechanism grants the slot to that connection with the greatest overall cost and then filters by layer. Here, connection A would be granted the slot. Now, assume that the layer 2 cell belongs to a broken GOB. Granting service to *A* will not improve the recipient's quality, and denying service to connections *B* and *C* potentially corrupts two additional GOBs. Denying service to A, while appearing to degrade QoS to the connection, actually provides a global benefit since connection *B* receives an additional scheduling opportunity.

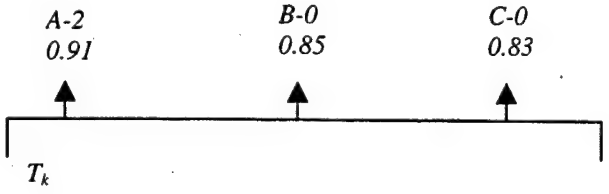

Figure VI.7: Competition Between Usable and Unusable Cells.

An analogous situation occurs with UBR connections carrying IP datagrams. If a cell from a datagram is discarded, the remaining cells are unusable. If the IP datagrams belong to a TCP connection, a single dropped cell forces the entire TCP segment to be retransmitted. As retransmissions reduce effective throughput, techniques such as partial packet discard respond to a dropped cell by dropping the remaining cells in the datagram. By removing these unusable cells from contention for scheduling, effective throughput is increased [18].

Based on the discussion above, we present a modification to the layered STEBR scheduling algorithm that implements partial GOB dropping. Partial GOB dropping consists of removing any cells remaining in a GOB following a dropped cell in that GOB. A similar approach proposed for high bandwidth MPEG-2 video traffic by Kuo and Ko [100] schedules slices for transmission only if sufficient bandwidth is available to transmit an entire slice without loss. The approach here is less stringent since scheduling assignments are made based on current queue occupancy, delay considerations do not allow determination of GOB length in real-time for low bit rate video traffic, and some partial benefit is derived by transmitting at least the beginning of the GOB.

Since the video stream is layered, partial GOB dropping must take into account both dropped cells within each GOB plus the impact of GOB corruption in one layer on related GOBs within other layers. Obviously, the greatest impact occurs when a GOB from the base layer is corrupted. In that case, at least part of the information carried within the associated GOBs of lower priority layers is also rendered unusable. Corruption of a lower priority GOB does not appear to have the same consequence. Based on subjective and quantitative evaluations using the coder from Chapter IV, a tangible benefit is obtained by decoding and applying a lower priority enhancement regardless of whether higher priority enhancement layers are successfully decoded.

Based on these observations, partial GOB dropping is implemented in the following manner. If a cell is discarded from a base layer GOB, all remaining cells in that GOB and all remaining cells in associated *lower* priority layer GOBs are discarded. If a cell is dropped from within an enhancement layer GOB, all remaining cells in that

layer's GOB are discarded. The impact of a cell discard in each situation is illustrated in Figure VI.8.

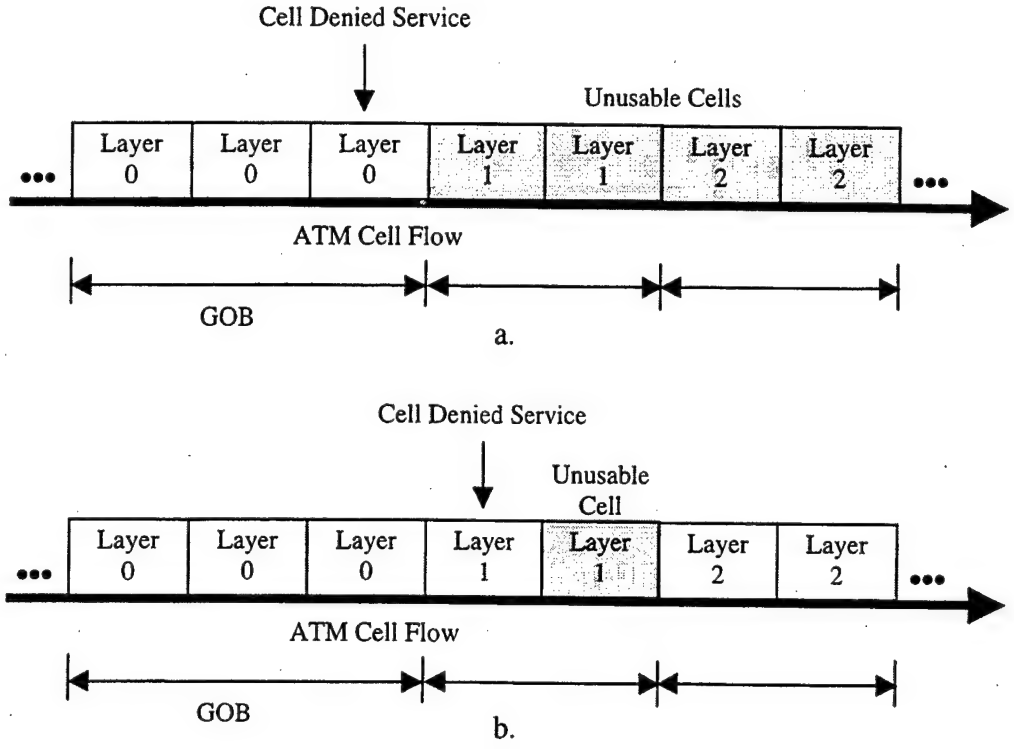

Figure VI.8: Discard Policy Following a Cell Loss from: a) Base Layer GOB or b) Enhancement Layer GOB.

The base layer discard policy is actually somewhat severe since a cell loss from a base layer GOB does not always invalidate information in enhancement layer GOBs. Technically a loss from the base layer GOB only invalidates information in enhancement layers starting at the same spatial position, i.e., a macroblock, for decoding purposes. Any information prior to this point is still usable although coordinating the spatial relationship of cells in different layers is not a trivial task. One possible approach is to interleave cells from different layers in a manner that approximates the correct spatial dependence such that when a cell from the base layer is dropped, loss of usable information is minimized when dropping the remaining cells in the base and enhancement layers. This approach is shown in Figure VI.9, where cells from the layer

GOBs have been interleaved due to spatial dependence. Now a loss of a base layer cell results in smaller number of cell discards compared to Figure VI.8. With the current coder, this is not an issue since at low bit rates the enhancement layers are usually restricted to a single ATM cell in length.

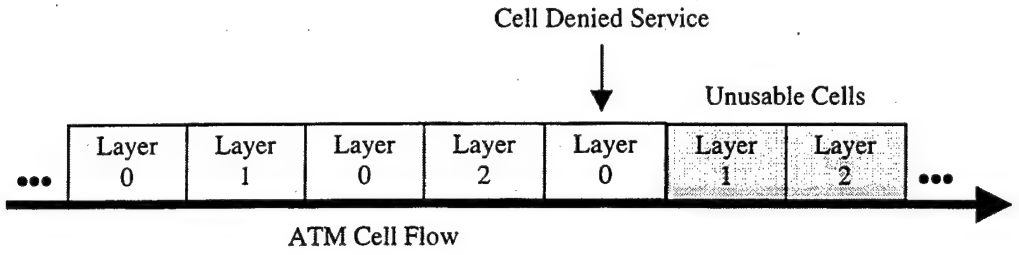

Figure VI.9: Interleaving Layers Cells to Minimize Information Loss.

GOB dropping is implemented in the following manner. For each connection, a flag is maintained for each layer, i.e., 3 flags per video source. The flag indicates the state of the current GOB in each layer, 'RETAIN' or 'DROP', and indicates whether the remaining cells in that GOB should be retained or dropped. Assuming that the current GOB has remained intact so far, a cell dropped due to expiration triggers a change in status from RETAIN to DROP. If the expired cell belongs to the base layer, the flags for the associated lower priority layer GOBs are also set to DROP. Each layer's flag is reset to RETAIN at the start of a new GOB as indicated by either the SDU bit or a change in cell tags (see Figure II.11 and Figure II.13).

At the start of each scheduling slot, the queue is scanned from head to tail as previously described. The scheduler performs different actions for each cell depending on the status of its parent GOB. If the GOB status is RETAIN, the cell's ToE is calculated. If the cell has expired, the cell is dropped, and the GOB's status is changed to DROP. Again, if the cell belonged to the base layer, the enhancement layers are set in a similar manner. If the GOB status is DROP, the cell is examined to determine if the cell contains a GOB header, which indicates the start of a new GOB. If it does and the cell has not expired, the GOB status is toggled back to GOOD. Otherwise, the cell is dropped regardless of its ToE. This algorithm is summarized in Figure VI.10.

1. Scan the queue from head to tail.
2. For each cell from connection i, layer j:

    A. If status[i,j] = RETAIN:
        a. Calculate ToE.
        b. If ToE < service time:
            i. Status[i,j] = DROP.
            ii. Discard cell.
            iii. If $j = 0$:
                1. Status[i,k] = DROP $\forall k \neq 0$.
    B. If status[i,j] = DROP:
        a. Check for GOB header.
        b. If new GOB header:
            i. Calculate ToE.
            ii. If ToE < service time:
                1. Discard cell.
            iii. Else:
                1. Status[i,j] = RETAIN.
        c. Else:
            i. Discard cell.

Figure VI.10: Partial GOB Dropping Algorithm.

D. RESULTS

Performance of the layered STEBR algorithm was validated using OPNET. The scenario simulated was a network configured as shown in Figure VI.11 with three layered video sources. Each layered source transmits at a mean bit rate of 80 kbps and is represented within the simulation using the MMRP traffic model discussed in Chapter V. An OPNET model for a layered video source is given in Appendix A, and the model parameters are given in Appendix B. The bit allocation among the layers was set at 2:1:1. The requested QoS for each connection consists of a maxCTD of 50 ms and a CLP of 10^{-3}. Each layer is assigned the same CLR. While the CLR is high for video traffic, the value chosen shortens simulation time while still giving a valid demonstration of the algorithm's behavior under different loads. Since the performance of the STEBR algorithm with heterogeneous traffic has been presented thoroughly elsewhere [39], only the homogenous traffic case is considered here.

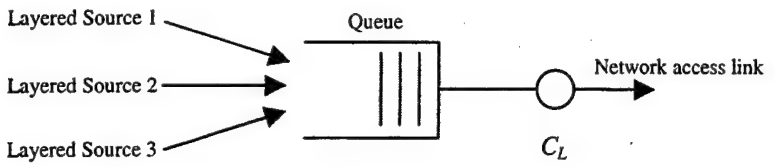

Figure VI.11: Network Scenario.

The first issue considered was the ability of the QoS filtering algorithms listed in Figure VI.4 to shift bandwidth to the higher priority base layer as network load was increased to simulate congestion and the corresponding impact on connection throughput. The first filtering approach considered was bandwidth sharing.

The premise of service deferral is sustaining the maximum possible throughput by deferring service of a selected cell provided that cell will not expire if not granted immediate service. Figure VI.12 shows the impact of service deferral on the CLR for each layer as network load is increased. As long as the base layer is receiving its required QoS, all layers are treated in approximately the same manner. As network load increases and connections experience CLRs exceeding the required CLR of 10^{-3}, the scheduler adapts by denying service to the higher layers whenever possible. However, with service deferral, cells from lower priority layers are still granted service unless a higher priority cell is present and about to expire. The result is that, while the scheduler violates QoS for the base layer last, QoS cannot be maintained over a wide range. Consequently, the gap in CLR between the base and enhancement layers stays relatively constant at one order of magnitude, and QoS between the enhancement layers is not differentiated at all.

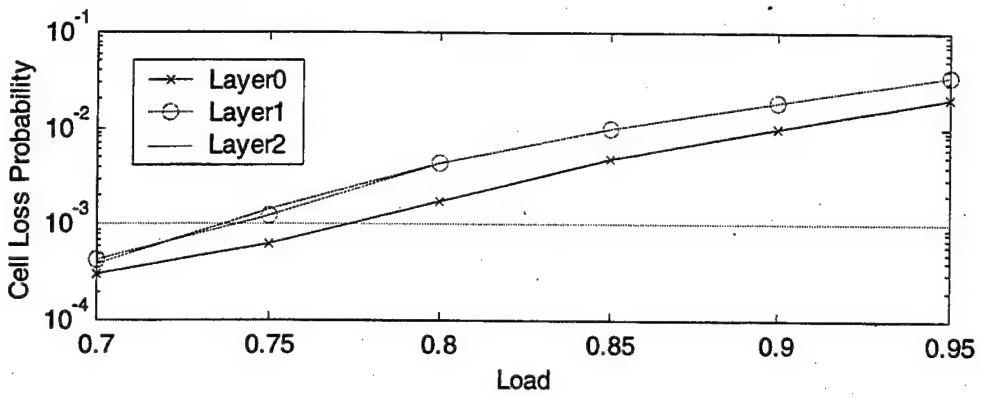

Figure VI.12: CLR for Bandwidth Sharing and Service Deferrals.

The same scenario without service deferral is shown in Figure VI.13. Now a higher priority cell receives priority service if the layer is not receiving its requisite QoS regardless of the cell's position within the queue. The result is that as network load is increased, the required QoS for the base layer is maintained regardless of network loading, and a clear delineation exits in treatment of the enhancement layers. Comparing Figure VI.13 with Figure VI.12, the bandwidth required to maintain QoS for the base layer comes primarily from denying service to layer 2 cells, the second enhancement layer as desired.

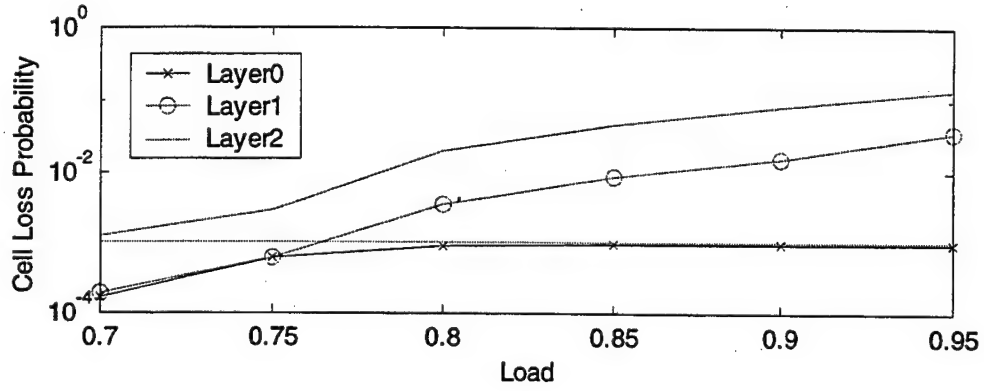

Figure VI.13: CLR for Bandwidth Sharing and No Service Deferrals.

The performance of priority sharing was also considered with and without service deferral. With service deferral, the result is identical to Figure VI.12. Service deferral renders the cost function irrelevant since the cost function is effectively applied only if

the chosen cell is about to expire. Without service deferral, the impact of priority sharing is shown in Figure VI.14. Since the base layer receives priority any time a cell is present, the scheduler actually prevents any observable cell loss in the base layer for the network loads examined. Once again, the bandwidth required comes at the expense of the second enhancement layer as desired. However, the first enhancement layer receives the best service out of all three scenarios.

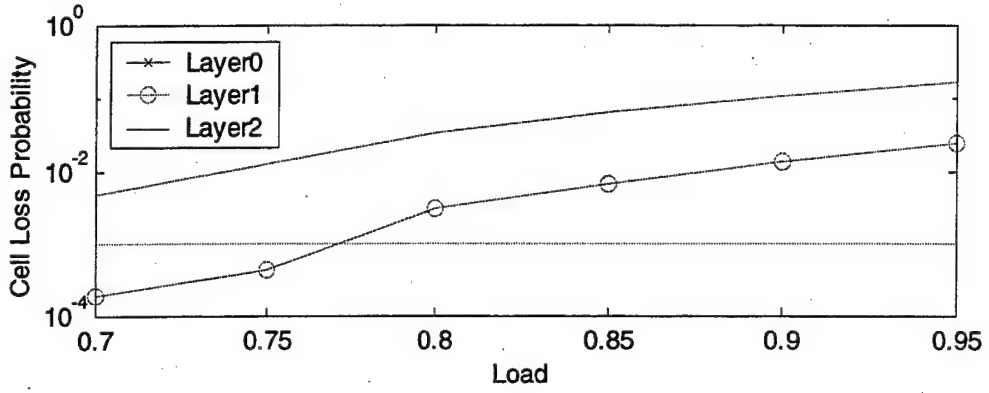

Figure VI.14: CLR for Priority Sharing and No Service Deferrals.

Comparing Figures VI.12 through VI.14, priority sharing without service deferrals gives the best performance with respect maintaining or exceeding the QoS for the layers in the hierarchical order desired. However, the QoS of the base layer cannot be arbitrarily controlled without impacting the connection's throughput. The throughput for each of the scenarios above is shown in Figure VI.15 and indicates that closer regulation of the CLR comes at the price of decreasing throughput for that connection. Given these results, the priority algorithm was deemed unsuitable. Since some loss can be tolerated in the base layer, as indicated by the QoS parameters supplied as part of the traffic contract, the priority sharing algorithm appears unsuitable. The remaining discussion covers only the bandwidth sharing algorithm.

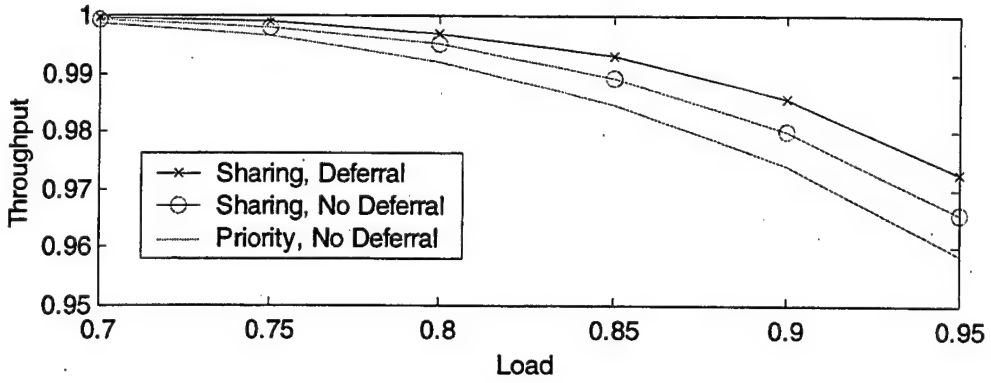

Figure VI.15: Throughput under Each Scheduling Scheme.

The next issue considered is the effect of partial GOB dropping for each of the two remaining scenarios. With GOB dropping, throughput is expected to decrease since at least part of the traffic allowed through will be unusable at the decoder. The results for bandwidth sharing and service deferral are shown in Figure VI.16. Compared to Figure VI.12, better performance is delivered in terms of CLR for each layer although the difference grows successively smaller with increasingly higher network loads. The improvement is the most notable with the second enhancement layer. Also a marked differentiation in QoS is observed for both of the enhancement layers that did not exist prior to GOB dropping.

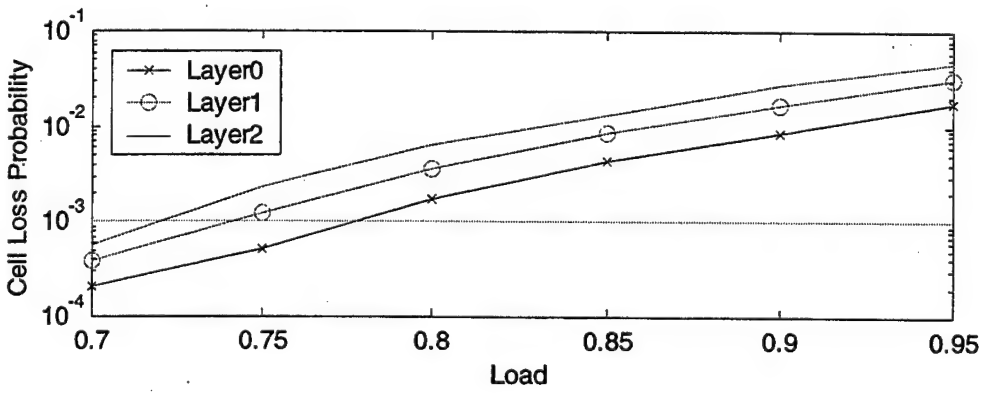

Figure VI.16: CLR for Bandwidth Sharing, Service Deferrals, and GOB Dropping.

The effect of GOB dropping without service deferrals is shown in Figure VI.17. The scheduler is still able to maintain the requisite CLR for the base layer. The effect on the enhancement layers is mixed. Control over the CLR for the first enhancement layer is improved relative to Figure VI.13 at network loads below 0.8. Above this point, CLR increases. The CLR for the second enhancement layer is higher regardless of the network load. The greater loss, however, results in the improved CLR observed for the first enhancement layer at lower network loads. At higher network loads, the impact of cell drops from the base layer dominates. Since a cell dropped from a base layer GOB causes the first and second layer's GOBs to be discarded, the CLRs for first and second enhancement layers start to converge.

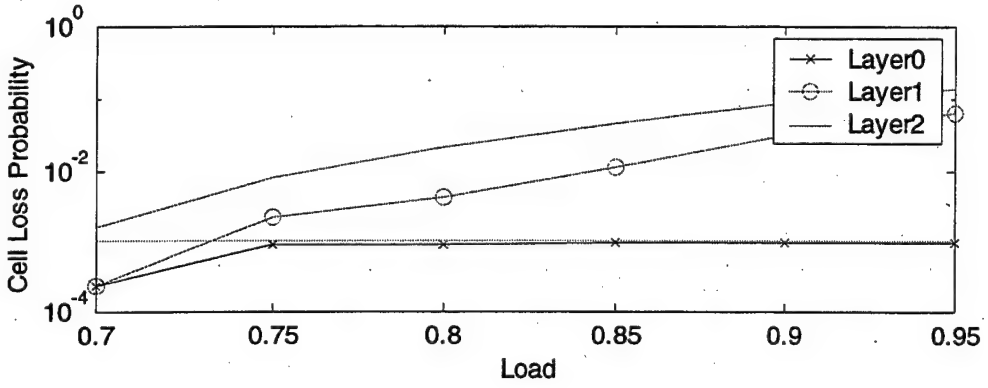

Figure VI.17: CLR for Bandwidth Sharing, No Service Deferrals, and GOB Dropping.

The impact of GOB dropping on throughput, with and without service deferral, is shown in Figure VI.18. Remarkably, in both cases, only a small decrease is observed in throughput and then only at high network loads. However, service deferrals still result in higher throughput overall.

Considering the joint effects of service deferral and GOB dropping on layered scheduling, the scheduler is able to more aggressively utilize bandwidth released by dropping non-viable cells to improve service for all layers. However, service deferral is unable to maintain the requisite CLR for the base layer at high network loads with or without GOB dropping. Without service deferral, throughput is impacted since the

scheduler gives greater priority to winning cells, which tend to be base layer cells at the higher loads. GOB dropping does allow the scheduler to reallocate bandwidth, at the expense of the second enhancement layer, to improve the CLR for the first enhancement layer at network loads below 0.8 and maintain the required service for the base layer.

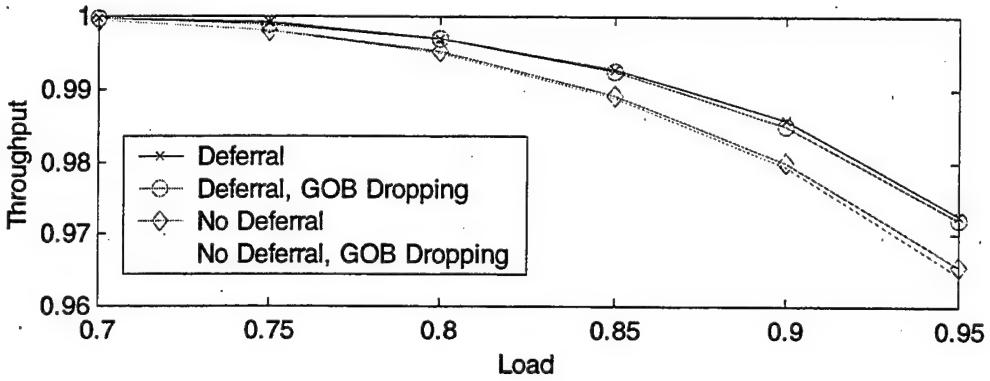

Figure VI.18: Throughput Variation with Partial GOB Dropping.

Comparing Figure VI.16 with Figure VI.17, service deferrals actually produce slightly better overall service, as demonstrated by lower CLR for the base layer and higher connection throughput, for network loads that maintain the base layer CLR below the target CLR. As network load increases, forgoing service deferrals results in better service to the base layer in terms of reduced CLR. These results suggest that the most effective scheduling scheme is actually a hybrid of the two approaches: use service deferrals when the base layer is receiving its required QoS and drop service deferrals when the base layer is not receiving its required QoS.

The final issue examined is how the cells with related GOBs are arranged within the cell flow. Each base layer GOB has two associated enhancement layer GOBs. The partial GOB dropping algorithm discards upper layer cells whenever a base layer cell is discarded. However, the number of cells actually discarded depends on how the cells from the individual layers are arranged, concatenated or interleaved in a manner that reflects the actual spatial dependency among the cells in the different layers as shown in Figure VI.9. The goal is to minimize information loss by only dropping those upper layer cells that are rendered unusable by a drop in the base layer.

To examine this idea, the bit allocation among the layers was increased to 4:2:2. While this is the same relative ratio considered earlier, each layer's GOB is now doubled in size to increase the effect of partial GOB dropping. Two arrangements were considered as shown in Figure VI.19. The first arrangement concatenates cells from each layer. The second arrangement interleaves the cells. In either case, base layer GOB headers occur every eight cells.

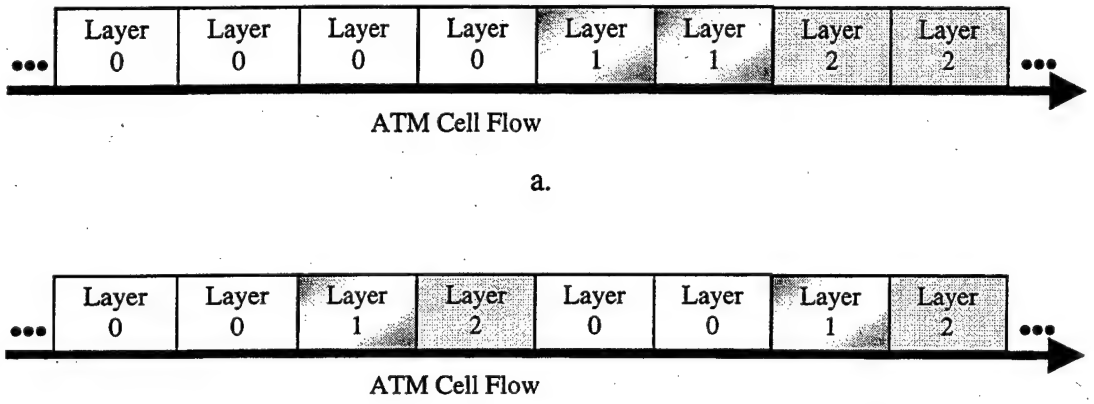

Figure VI.19: Cell Arrangements Considered for a 4:2:2 Bit Allocation: a) Concatenated or b) Interleaved.

The effect of each cell arrangement using bandwidth sharing and service deferrals is shown in Figure VI.20. For the base and first enhancement layers, interleaving cells from different layer GOBs improves CLR consistently regardless of the network load. Not surprisingly, the improved CLR comes at the expense of higher CLR for the second enhancement layer over the range of network loads examined. However, performance is judged unacceptable since, although a clear differentiation in service exists for each layer, QoS degrades for each layer at approximately the same rate instead of favoring the base layer at the higher network loads. Throughput differences for each case were negligible.

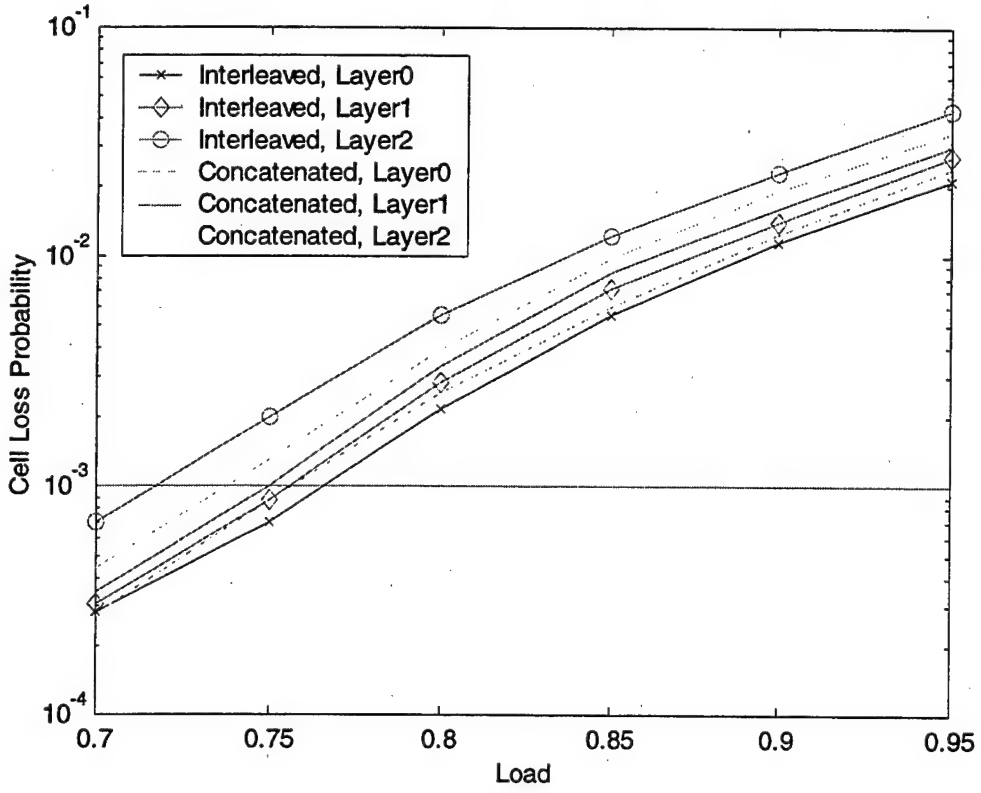

Figure VI.20: Relative Affect of Interleaving and Concatenating on CLR with Bandwidth Sharing and Service Deferrals.

The effect of each cell arrangement using bandwidth sharing and service deferrals is shown in Figure VI.21. Interleaving gives a similar performance benefit to the one discussed in the last paragraph. CLRs are improved for both the base and first enhancement layers. There are two notable distinctions between concatenating and interleaving. As observed previously, forgoing service deferrals allow the scheduler to maintain the requisite QoS for the base layer. By concatenating or interleaving, the same is observed on Figure VI.21. However, interleaving still improves CLR by a small measure at each network load examined. For the first enhancement layer, unlike previous simulations, interleaving allows QoS to be maintained up to a network load of 0.8 although it increases rapidly beyond this point. Also notable is the observation that interleaving improves the CLR for the second enhancement layer up to network loads

exceeding 0.8 although performance degrades relative to concatenation after this point. Again, throughput differences for each case were negligible.

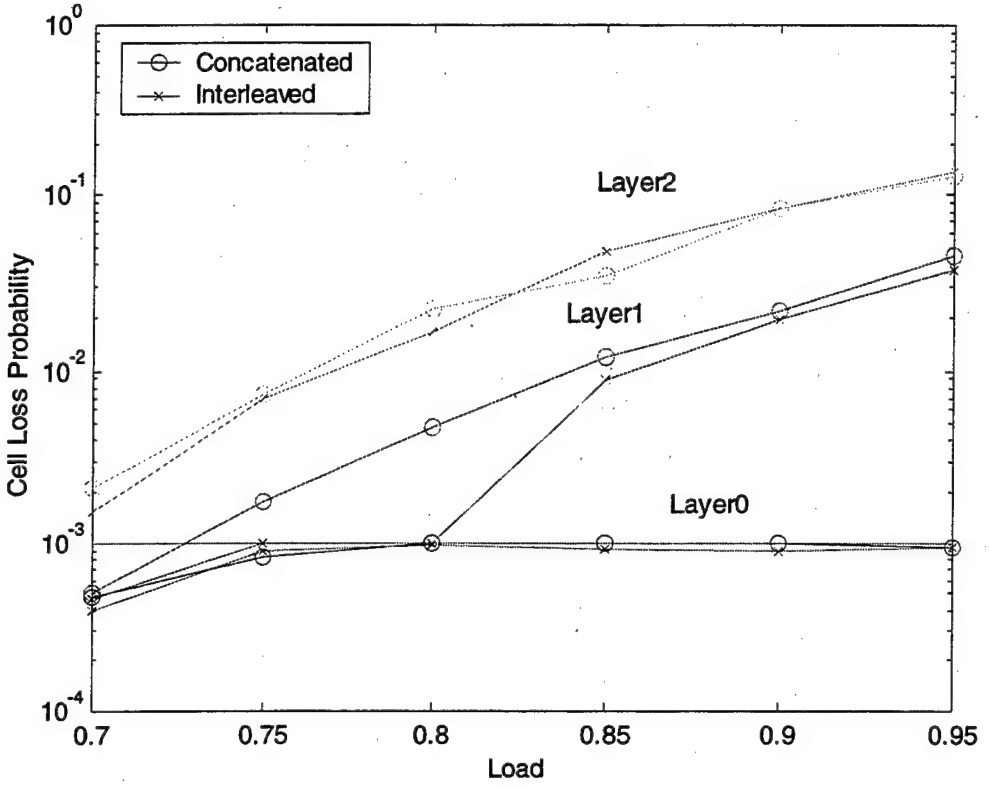

Figure VI.21: Relative Affect of Interleaving and Concatenating on CLR with Bandwidth Sharing and Without Service Deferrals.

This chapter presented a scheduling algorithm for layered video traffic based on the STEBR algorithm originally proposed by Uziel [39]. The STEBR algorithm provides optimal scheduling for heterogeneous traffic, where each connection possibly has different CLR and CTD requirements. The hierarchical nature of layered video is introduced through a prioritization scheme that denies service to cells from lower priority layers during periods of congestion, thereby increasing the probability that cells from higher priority layers are transmitted. In this manner, the quality of the reconstructed video degrades in a graceful manner than if cells were dropped indiscriminately from the

connection. Effective throughput is increased through partial GOB dropping which also drops cells determined to be unusable to the decoder. Dropping these cells increases scheduling opportunities for viable cells and increases the probability of transmitting higher priority GOBs intact.

VII. CONCLUSIONS

A. SUMMARY OF WORK

Motivated in part by the US Navy's IT-21 initiative, there has been considerable interest in deploying multimedia applications over tactical networks. Tactical networks may be characterized as low bit rate, unreliable, and heterogeneous. Multimedia applications, especially those incorporating video, tend to be bandwidth intensive and sensitive to transmission errors. Traditional multimedia processing techniques do not take these constraints into account.

This work investigated issues related to distributing low-bit-rate video within the context of a teleconferencing application deployed over a tactical ATM network. The main objective was to develop mechanisms that support transmission of low-bit-rate video streams as a series of scalable layers that progressively improve quality. These mechanisms exploit the hierarchical nature of the layered video stream along the transmission path from the sender to the recipients to facilitate transmission. Specifically, the approach proposed in this dissertation works across the application-network interface by coding the video stream into layers, shaping the resulting layered video stream prior to entry into the network, and prioritizing service in accordance with the relative perceptual importance of each layer.

A new layered video coding scheme was developed that includes a number of original contributions. This work codified some of the design issues required for an effective layered coder. How to layer the video stream *effectively* is an elementary design issue. To address this, a series of heuristic rules were proposed that lead to effective layering structures for motion video via wavelet-based subband decomposition. These rules stem from a simple split-and-merge algorithm that uses subband variance as a measure of perceptual relevance. By grouping subbands of like variance and assigning subbands to layers in order of perceptual importance, the video stream is divided into the requisite number of layers. We applied this heuristic rule set and devised a three-layer

coding scheme for low-motion video. Employing a common layering scheme for both motion video and static presentation slides yielded poor results due to their different energy distributions among the subbands and differing perceptual weighting of high frequency content. Consequently, we devised a separate scheme in which each layer incorporates contributions from all frequency bands.

A new suboptimal rate control scheme for layered video was developed. Using classical rate-distortion approaches, constraining the bit rate for a layered video stream using k quantizers involves simultaneously solving k cost functions. In this work, a simpler approach replaced the k-dimensional rate-distortion problem with a one-dimensional operational rate-distortion curve generated from a set of suboptimal quantizer vectors. Rate control is then implemented via a table lookup into a codebook containing the suboptimal quantizer vectors.

The effect of traffic smoothing, prior to network entry, on queuing performance and scheduling efficiency was examined. The approach investigated smoothing at three time scales: frame, layer, and cell interarrival. Smoothing at the frame level is performed by the rate controller and requires no special implementation. Smoothing within the frame is accomplished using a leaky-bucket mechanism whose token rate changes each frame. Implementations were proposed for transmitting layers over a single VCI and multiple VCIs as well as the implications of positioning the leaky bucket prior to the ATM layer.

The problem of prioritizing cell scheduling in layered video traffic was investigated to enable a more graceful degradation in received video quality during periods of high cell loss. QoS at the connection level is maintained using the STEBR algorithm originally proposed by Uziel [39]. Within the connection, a prioritization scheme denies service to cells from lower priority layers as required to maintain the requisite QoS, in terms of cell loss rate, for higher priority layers are transmitted. This ensures that reconstructed video quality degrades more gracefully than if cells were dropped indiscriminately from the connection. Since the decoder resynchronizes using GOB headers following data loss, a cell dropped within a GOB renders any remaining

cells in the GOB unusable. We proposed partial GOB dropping to increase effective throughput by intelligently discarding related cells deemed unusable that would otherwise compete for and waste scheduling opportunities.

Scheduling at the layer level, in addition to the connection level, requires a means for associating cells with layers. Also, partial GOB dropping requires the scheduler to have the ability to identify GOB headers within each layer. Two approaches were considered. The first approach assigns each layer to a separate VCC using AAL5. This approach is the simplest in terms of implementation but requires increased signaling in multicast scenarios. The second approach multiplexes each layer across a single VCC using AAL2. This approach offers quicker call establishment and minimizes signaling in multicast scenarios but requires modification to the CPCS sublayer and does not scale beyond four layers.

B. SUGGESTIONS FOR FUTURE RESEARCH

The coder as proposed in Chapter IV supports only 8-bit grayscale video. Extension to 24-bit color video is a natural step in the maturation of the coder design. Video capture usually results in three bit planes, a luminous plane and two color difference planes, each with the same resolution as the original frame. Since the HVS is more sensitive to variations in luminosity than color, the color planes are normally subsampled relative to the luminous plane [6]. With 4:2:2 subsampling, each 16×16 macroblock in the original frame is represented as a 16×16 luminance macroblock and two 8×8 color difference macroblocks. The work presented in Chapter IV applies only to the luminance portion of the picture. More research is required to investigate a general layering structure for the color difference components. While the frequency characteristics of the color components might be expected to mirror those of the luminance components, the perceptual importance of those components clearly does not. In the quantization matrix suggested for the color components by the JPEG standard, little discrimination is made between low and high frequencies, between vertical and

horizontal detail [66]. Whether a separate approach is required for the color content of static slides also bears consideration.

One area not fully exploited by the proposed coder is recent advancements in entropy coding. One promising area of research is the concept of reversible codes, i.e., codes that are uniquely decipherable by parsing forward or backward through the bitstream. With a reversible code, the decoder would respond to a stream interruption by buffering the bitstream until the next GOB header is located. Then the decoder could parse backwards to recover a portion of the corrupted GOB. An interesting analysis could focus on the relative benefits of reversible coding and partial GOB dropping since the two approaches could not coexist.

Other issues concerning the coder design that were only partially investigated include rate control at the macroblock level and error concealment schemes at the decoder. The results presented in Chapter IV only incorporate rate control at the frame level in which the quantizer vector is changed solely at the beginning of each new frame. Tighter control is possible by implementing rate control at the macroblock level and allowing the quantizer vector to change within the frame. The issue is whether changes to the quantizer vector within the frame would be distinctly perceptible. The final coder issue is implementing error concealment at the decoder. The decoder may use error concealment to compensate for incomplete information when reconstructing a frame. A simple but effective technique implemented here is zeroth order error concealment. If the decoder cannot determine if a macroblock should have been updated, the corresponding macroblock in the last frame is used by default. This is particularly effective with low motion video. More aggressive approaches to consider would employ prediction or interpolation to estimate missing coefficients from adjacent macroblocks.

The MMRP model appears quite effective at representing VBR video, and the associated queuing analysis tools are mature. However, the approach recommended by Skelly et al. [14] uniformly quantizes the video stream. Experimentally determined histograms demonstrate that video, regardless of the motion content, is decidedly non-uniform in distribution [27]. Since MMRP queuing techniques stem from an estimate of

the bit rate distribution, an accurate representation of this distribution is essential. Given that video is not distributed uniformly, non-uniform quantization schemes bear examination to improve the representation for a given number of states. One approach is to use Max-Lloyd quantizers [6], or an optimal representation could be developed directly from the original histogram.

The STEBR algorithm provides a powerful, optimal scheduling algorithm for CBR and VBR real-time traffic with constraints on CLR and CTD. Two extensions appear worth further consideration to extend the algorithm. First, the STEBR algorithm makes scheduling decisions based on the past history of each connection and the current queue state *assuming* that no further arrivals take place during the current scheduling slot. A possible extension is to modify the cost function to consider the impact of predicted near-term arrivals for each connection. Predicting future arrivals requires that the scheduler maintain a suitable traffic model for each connection or an aggregate of related connections. Modeling bursty sources appears difficult in the context of real-time scheduling decisions, as opposed to buffer sizing, but predicting the behavior of multiplexed traffic, as typified by the approach taken for VBR video in [95], may prove feasible.

Another worthwhile extension to STEBR is to incorporate the UBR and ABR service categories to create a uniform optimal scheduling algorithm. As STEBR is cost-based, extension requires construction of a cost-function suitable for each service category. For example, UBR connections can be assigned a permanent cost of one, thus restricting service unless all other connections are receiving their required QoS. Such an assignment appears suitable since UBR connections are assigned unutilized bandwidth from CBR and VBR connections. A suitable cost function for ABR is the ratio of MCR to instantaneous cell rate granted by the scheduler. However, ABR throughput benefits from employing feedback to regulate the sender's transmission rate both to match available bandwidth and to fairly share available bandwidth among all the active ABR connections. A scheme for incorporating these mechanisms into the STEBR algorithm requires additional consideration.

APPENDIX A. OPNET MODEL CODE

This appendix contains the OPNET process models used to generate the simulations results shown in Chapters V and VI. Each process model consists of a finite state machine and a series of code segments that implement the behavior required for each state.

A. LAYERED VIDEO SCHEDULER

The OPNET model for the layered scheduler implements the layered scheduling algorithm discussed in Chapter VI. Specifically, STEBR is used to select the winning connection at the beginning of each service interval, and the CLPRs for each layers are filtered and compared to determine the winning layer. The code also implements partial GOB dropping as an option. The scheduler assumes that each source transmits using only a single VCC (see Figure II.12). While the code is specifically tailored for the homogeneous traffic case, the model is easily extended to heterogeneous traffic by storing the connection type with the connection's VCC and performing QoS filtering if the connection is carrying layered video. The finite state machine is shown in Figure A.1.

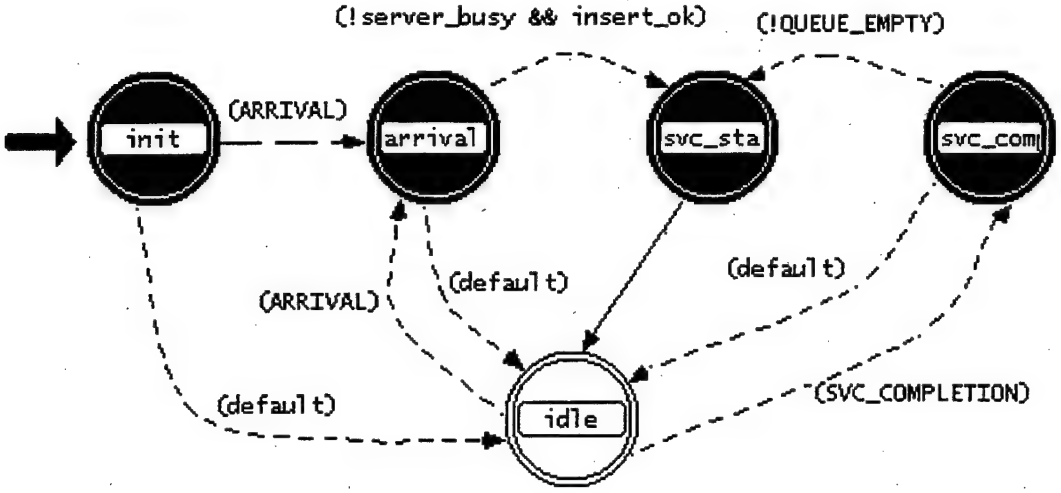

Figure A.1: Finite State Machine for Scheduler Process Model.

1. Header Block

```
#include  "ams_pk_support.h"
#include  <math.h>

#define QUEUE_EMPTY      (op_q_empty ())
#define SVC_COMPLETION   op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL          op_intrpt_type () == OPC_INTRPT_STRM

#define VCI_BASE         100
#define MAX_SOURCE       7
#define MAX_LAYER        3

#define DROP             1
#define RETAIN           0
#define NEWHEADER        1

void order_queue(int);
int expire_cells(void);
```

2. State Variable Block

```
int          \server_busy;
double       \service_rate;
Objid        \own_id;
Stathandle   \clp_handle;
int          \cell_count[MAX_SOURCE];
int          \layerCellCount[MAX_SOURCE][MAX_LAYER];
int          \cells_dropped[MAX_SOURCE];
```

```
int          \layerCellsDropped[MAX_SOURCE][MAX_LAYER];
double       \clp;
double       \pk_svc_time;
Stathandle   \cell_handle;
int          \cells_serviced;
Stathandle   \util_handle;
double       \maxCTD;
double       \maxCLP;
int          \cells_waiting[MAX_SOURCE];
int          \gobDrop[MAX_SOURCE][MAX_LAYER];
Stathandle   \clpr0_handle;
Stathandle   \clpr1_handle;
Stathandle   \clpr2_handle;
```

3. Temporary Variable Block

```
Packet*          pkptr;
int              insert_ok;
int              num_cells;
int              ix,jx;
int              source_id;
int              layer_id;

AtmT_Cell_Header_Fields*
                 atm_hdr_ptr;
int              total_arrived;
int              total_dropped;
double           minToE;
int              cell_to_send;
double           iCLP;
double           clpr[MAX_SOURCE];
double           delta[MAX_SOURCE];
double           max_clpr;
int              winner;
int              winningLayer;

int              extra_cells[MAX_SOURCE];
double           cost[MAX_SOURCE];
int*             service_slot;
int*             slotSourceID;
double*          cell_cost;
int              q_index;
int              slot;
int              max_index;
double           max_cost;
int              done;

double           iLayerCLP;
double           layerCLPR[MAX_SOURCE][MAX_LAYER];
int              layerCellsWaiting[MAX_SOURCE][MAX_LAYER];
double           filteredCost[MAX_LAYER];
double           filteredCLPR[MAX_LAYER];
double           max_CLPR;
```

4. Function Block

expire_cells() removes cell from the queue that have expired or as required by the partial GOB dropping algorithm. With partial GOB dropping, a flag indicates the status for each layer within a connection. An expired cell toggles the flag to "DROP". If the expired cell belongs to the base layer, flags are set to "DROP" for each of the other layers. GOB headers reset the flags to "RETAIN". Partial GOB dropping may be disabled by commenting out the lines highlighted in **bold**.

```
int expire_cells(void)
{
   int num_cells;
   int source_id;
   int layer_id;
   int gobHeader;
   int ix;
   Packet* pkptr;
   AtmT_Cell_Header_Fields* atm_hdr_ptr;

   /* Find the number of cells in the queue. */
   num_cells = op_subq_stat(0, OPC_QSTAT_PKSIZE);

   /* Remove cells that cannot complete service before expiring,
starting at the */
   /* tail of the queue. */
   ix = 0;
   while (ix < num_cells){
      pkptr = op_subq_pk_access(0, ix);
      op_pk_nfd_get(pkptr, "header fields", &atm_hdr_ptr);
      source_id = (atm_hdr_ptr->VCI - VCI_BASE);
      layer_id = atm_hdr_ptr->PT + 2*atm_hdr_ptr->CLP;
      gobHeader = atm_hdr_ptr->GFC;

      if (gobDrop[source_id] == DROP){
         if (gobHeader == NEWHEADER){
            if ((maxCTD - op_q_wait_time(pkptr)) < pk_svc_time){
               pkptr = op_subq_pk_remove(0, ix);
               op_pk_destroy(pkptr);
               op_prg_mem_free(atm_hdr_ptr);
               cells_dropped[source_id]++;
               layerCellsDropped[source_id][layer_id]++;
               num_cells--;
               cells_waiting[source_id]--;
            }
            else{
               if (layer_id == 0){
               gobDrop[source_id][0] = RETAIN;
```

```
                gobDrop[source_id][1] = RETAIN;
                gobDrop[source_id][2] = RETAIN;
                }
                else{
                   if (gobDrop[source_id][0] == RETAIN){
                      gobDrop[source_id][layer_id] = RETAIN;
                   }
                }
                /* Reload the header field struct. */
                op_pk_nfd_set(pkptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
                   op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));
                ix++;
             }
          }
          else{
             pkptr = op_subq_pk_remove(0, ix);
             op_pk_destroy(pkptr);
             op_prg_mem_free(atm_hdr_ptr);
             cells_dropped[source_id]++;
             layerCellsDropped[source_id][layer_id]++;
             num_cells--;
             cells_waiting[source_id]--;
          }
       }
       else if ((maxCTD - op_q_wait_time(pkptr)) < pk_svc_time){
          pkptr = op_subq_pk_remove(0, ix);
          op_pk_destroy(pkptr);
          op_prg_mem_free(atm_hdr_ptr);
          cells_dropped[source_id]++;
          layerCellsDropped[source_id][layer_id]++;
          num_cells--;
          cells_waiting[source_id]--;
          gobDrop[source_id][layer_id] = DROP;
          if (layer_id == 0){
             gobDrop[source_id][1] = DROP;
             gobDrop[source_id][2] = DROP;
          }
       }
       else{
       /* Reload the header field struct. */
       op_pk_nfd_set(pkptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
             op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));
       ix++;
       }
    }//while

    return num_cells;
}
```

order_queue() reorders the queue in order of increasing ToE from the head of the queue.

```
void order_queue(int num_cells)
{
   double *ToE;
   double temp;
   int ix,jx;
   int sorted;
   Packet* pkptr;

   /* Allocate memory for array consisting of ToE entries. */
   ToE = (double*) op_prg_mem_alloc(num_cells*sizeof(double));

   /* Parse the queue and determine each cell's ToE. */
   for (ix = 0;ix < num_cells;ix++){
      pkptr = op_subq_pk_access(0, ix);
      ToE[ix] = maxCTD - op_q_wait_time(pkptr);
   }

   /* Queue is originally unsorted. */
   sorted = OPC_FALSE;

   /* Perform a bubble sort. */
   for (ix = 0; !(sorted) && ix < (num_cells - 1); ix++){
      sorted = OPC_TRUE;

      for (jx = 0;jx < (num_cells - ix - 1);jx++){
         if (ToE[jx] > ToE[jx+1]){
            temp = ToE[jx];
            ToE[jx] = ToE[jx+1];
            ToE[jx+1] = temp;
            op_subq_pk_swap(0,jx,jx+1);
            sorted = OPC_FALSE;
         }
      }
   }

   /* Free the memory. */
   op_prg_mem_free(ToE);
}
```

5. Init State

The Init State initializes all statistics and counters and sets the QoS parameters required for each connection. Since only homogenous traffic is considered, only a single set of parameters is listed.

```
/* initially the server is idle */
server_busy = 0;

/* get queue module's own object id */
own_id = op_id_self ();

/* get assigned value of server processing rate */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);

pk_svc_time = 1.0 / service_rate;

/* Declare local statistics. */
clp_handle = op_stat_reg("CLP",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);
cell_handle = op_stat_reg("Time",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);
util_handle =
op_stat_reg("Utilization",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);
clpr0_handle = op_stat_reg("CLPR0",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);
clpr1_handle = op_stat_reg("CLPR1",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);
clpr2_handle = op_stat_reg("CLPR2",OPC_STAT_INDEX_NONE,OPC_STAT_LOCAL);

for (ix=0; ix < MAX_SOURCE;ix++){
   cell_count[ix] = 0;
   cells_dropped[ix] = 0;
   cells_waiting[ix] = 0;
   gobDrop[ix] = RETAIN;
   for (jx=0;jx < MAX_LAYER;jx++){
      layerCellCount[ix][jx] = 0;
      layerCellsDropped[ix][jx] = 0;
   }
}
cells_serviced = 0;

op_stat_write(cell_handle,(double)cell_count[0]);

/* Declare the QoS parameters. */
maxCTD = 0.050;
maxCLP = 0.001;
```

6. Arrival State

The Arrival State acquires arriving cells and updates the connection statistics. Each cell arrival also triggers recording of the CLP QoS statistic.

```
/* acquire the arriving packet              */
/* multiple arriving streams are supported. */
pkptr = op_pk_get (op_intrpt_strm ());

/* Get the source ID from the VCI and increment arrival count for the
source and layer. */
op_pk_nfd_get(pkptr, "header fields", &atm_hdr_ptr);
source_id = (atm_hdr_ptr->VCI - VCI_BASE);
```

```
layer_id = atm_hdr_ptr->PT + 2*atm_hdr_ptr->CLP;
cell_count[source_id]++;
layerCellCount[source_id][layer_id]++;

/* Reload the header field struct. */
op_pk_nfd_set(pkptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
             op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));

/* attempt to enqueue the packet at tail of subqueue 0 */
if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
    {
    /* the insertion failed (due to a full queue) */
    /* deallocate the packet */
    op_pk_destroy (pkptr);
    cells_dropped[source_id]++;
    layerCellsDropped[source_id][layer_id]++;

      /* set flag indicating insertion fail */
      /* this flag is used to determine transition */
      /* out of this state */
   insert_ok = 0;
    }
else{
      /* insertion was successful */
    insert_ok = 1;
    cells_waiting[source_id]++;
    }

// Capture connection statistics.
total_arrived = 0;
total_dropped = 0;
for (ix=0; ix < MAX_SOURCE; ix++){
   total_arrived += cell_count[ix];
   total_dropped += cells_dropped[ix];
}
clp = ((double)total_dropped)/total_arrived;

if (op_sim_time() > 0.0){
   op_stat_write(clp_handle,clp);
}

if (op_sim_time() > 0.0){
   op_stat_write(cell_handle,((double)total_arrived)/op_sim_time());
}

if (layerCellCount[1][0] > 0){
op_stat_write(clpr0_handle,((double)layerCellsDropped[1][0])/layerCellC
ount[1][0]);
}
if (layerCellCount[1][1] > 0){
op_stat_write(clpr1_handle,((double)layerCellsDropped[1][1])/layerCellC
ount[1][1]);
}
```

```
if (layerCellCount[1][2] > 0){
op_stat_write(clpr2_handle,((double)layerCellsDropped[1][2])/layerCellC
ount[1][2]);
}
```

7. SVC_Start State

The SVC_Start state determines which cell to process after removing expired cells and discarding cells from corrupted GOBs. STEBR determines the winning connection and the winning layer is determined after cost filtering. Service deferral is optional. Code segments highlighted in **bold** text indicate where cost-filtering algorithm can be altered and where service deferral may be activated.

```
/* In this state, at least one cell may require service.  Find the
number of cells. */
num_cells = expire_cells();

/* Sort the queue in descending order of ToE from the tail of the
queue. */
if (num_cells >0){
order_queue(num_cells);
}

/* Update the CLP ratios and the delta cost. */
for (ix=0; ix < MAX_SOURCE;ix++){

   iCLP = 0.0;
   delta[ix] = 0.0;
   if (cell_count[ix] > 0){
      iCLP = ((double)cells_dropped[ix])/cell_count[ix];
      delta[ix] = 1.0 /(cell_count[ix] * maxCLP);
   }
   clpr[ix] = iCLP/maxCLP;

   /* Update the layer statistics. */
   for (jx = 0;jx < MAX_LAYER;jx++){
      iLayerCLP = 0.0;
      if (layerCellCount[ix][jx] > 0){
         iLayerCLP =
((double)layerCellsDropped[ix][jx])/layerCellCount[ix][jx];
      }
      layerCLPR[ix][jx] = iLayerCLP/maxCLP;
   }
}


/* Initialize the connection cost and extra cell counts. */
for (ix=0; ix < MAX_SOURCE;ix++){
```

```
    extra_cells[ix] = 0;
    cost[ix] = -1.0;

    /* Initialize the current layer count. */
    for (jx = 0;jx < MAX_LAYER;jx++){
       layerCellsWaiting[ix][jx] = 0;
    }
}


/* Determine the current service slots and cost of each cell in the
queue. */
if (num_cells > 0){
   cell_cost = (double*) op_prg_mem_alloc(num_cells*sizeof(double));
   service_slot = (int*) op_prg_mem_alloc(num_cells*sizeof(int));
   slotSourceID = (int*) op_prg_mem_alloc(num_cells*sizeof(int));
}

for (ix = 0;ix < num_cells;ix++){
   pkptr = op_subq_pk_access(0,ix);
   service_slot[ix] = (int)floor((maxCTD -
op_q_wait_time(pkptr))/pk_svc_time);
   op_pk_nfd_get(pkptr, "header fields", &atm_hdr_ptr);
   source_id = atm_hdr_ptr->VCI - VCI_BASE;
   layer_id = atm_hdr_ptr->PT + 2*atm_hdr_ptr->CLP;
   slotSourceID[ix] = source_id;
   layerCellsWaiting[source_id][layer_id]++;
   op_pk_nfd_set(pkptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
              op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));
   clpr[source_id] += delta[source_id];
   cell_cost[ix] = clpr[source_id];
}

// STEBR starts here!

/* Grant service! */
if (num_cells > 0){

   /* Work from tail of queue forward to head. */
   q_index = num_cells - 1;
   done = OPC_FALSE;

   for (slot = service_slot[num_cells-1];(slot > 0) && (done !=
OPC_TRUE);slot--){

      /* Examine cells in the current time slot. */
      while ((q_index >= 0) && (service_slot[q_index] == slot)){
         source_id = slotSourceID[q_index];
         layer_id = slotLayerID[q_index];

         /* Cost out the source. */
         if (cost[source_id] >= 0){
            extra_cells[source_id]++;
```

```
            }
            else{
               cost[source_id] = cell_cost[q_index];
            }

            q_index--;
         }

         /* Determine which connection is granted service in current slot
*/
         /* based only on connection costs.
*/
         max_cost = cost[0];
         max_index = 0;
         for (ix = 1;ix < MAX_SOURCE;ix++){
            if(cost[ix] > max_cost){
               max_cost = cost[ix];
               max_index = ix;
            }
         }

         /* Assign the source to this slot if there are cells available.
*/
         if (cost[max_index] >= 0){
            winner = max_index;

            // Source has only one cell in the interval.
            if (extra_cells[max_index] == 0){
               cost[max_index] = -1;
            }
            // Source has more than one cell in the interval.
            else{
               extra_cells[max_index]--;
               cost[max_index] = cost[max_index] - delta[max_index];

               // Load the layer costs.
               for (ix = 0;ix < MAX_LAYER;ix++){
                  filteredCost[ix] = layerCost[winner][ix];
               }
            }
         }
         else if (q_index < 0){
            done = OPC_TRUE;
         }

      }//for

      /* Locate a cell from the winning source. */

      /* Prune the costs of the winning source. */
      for (jx = 1;jx < MAX_LAYER; jx++){
         if (layerCellsWaiting[winner][jx] == 0){
            layerCLPR[winner][jx] = 0;
         }
```

```
    }

    /* Find the winning layer from the source. */
    for (ix = 0;ix < MAX_LAYER;ix++){
       filteredCLPR[ix] = layerCLPR[winner][ix];
    }

    /* Filter the CLPR's to emphasize lower layers. */
    if (filteredCLPR[0] > 1.0){
       filteredCLPR[1] = 0.0;
       filteredCLPR[2] = 0.0;
    }
    else if (filteredCLPR[1] > 1.0){
       filteredCLPR[2] = 0.0;
    }

    /* Pick the layer with highest CLPR. */
    winningLayer = 0;
    max_CLPR = filteredCLPR[0];
    for (ix = 1;ix < MAX_LAYER;ix++){
       if (filteredCLPR[ix] > max_CLPR){
          winningLayer = ix;
          max_CLPR = filteredCLPR[ix];
       }
    }

    cell_to_send = 0;
    for (ix = 0;ix < num_cells;ix++){
       if ((slotSourceID[ix] == winner) && (slotLayerID[ix] ==
winningLayer)){
          cell_to_send = ix;
          break;
       }
    }

    //Activate service deferral here.
/*
    if (service_slot[cell_to_send] > 2){
       for (ix = 0;ix < num_cells;ix++){
          if (slotSourceID[ix] == winner){
             cell_to_send = ix;
             break;
          }
       }
    }
*/

    // Bubble the cell to head of the queue.
    if (cell_to_send > 0){
       for (ix = cell_to_send;ix > 0;ix--){
          op_subq_pk_swap(0,ix,ix-1);
       }
    }
```

```
    // Grant service to the cell.
    op_intrpt_schedule_self(op_sim_time() + pk_svc_time,0);

    // The server is now busy.
    server_busy = 1;
}

/* Free memory. */
if (num_cells >0){
    op_prg_mem_free(cell_cost);
    op_prg_mem_free(service_slot);
    op_prg_mem_free(slotSourceID);
}
```

8. SVC_Complete State

The SVC_Complete State removes a packet from the queue that has finished transmission.

```
/* Cell at the head of the queue */
/* is just finishing service */
pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

/* Update the source cells waiting count. */
op_pk_nfd_get(pkptr, "header fields", &atm_hdr_ptr);
source_id = (atm_hdr_ptr->VCI - VCI_BASE);
op_prg_mem_free(atm_hdr_ptr);
cells_waiting[source_id]--;

/* forward the packet on stream 0, */
/* causing an immediate interrupt at dest. */
op_pk_send_forced (pkptr, 0);

/* server is idle again. */
server_busy = 0;
```

B. LAYERED VIDEO SOURCE

The layered video process model represents up to *N* layered video source using a six-state MMRP with a deterministic arrival process. Cells from each layer of a particular source are multiplexed over a single VCI. Therefore, each cell is tagged using the scheme shown in Table II.3 to identify its parent layer. The finite state machine is shown in Figure A.2.

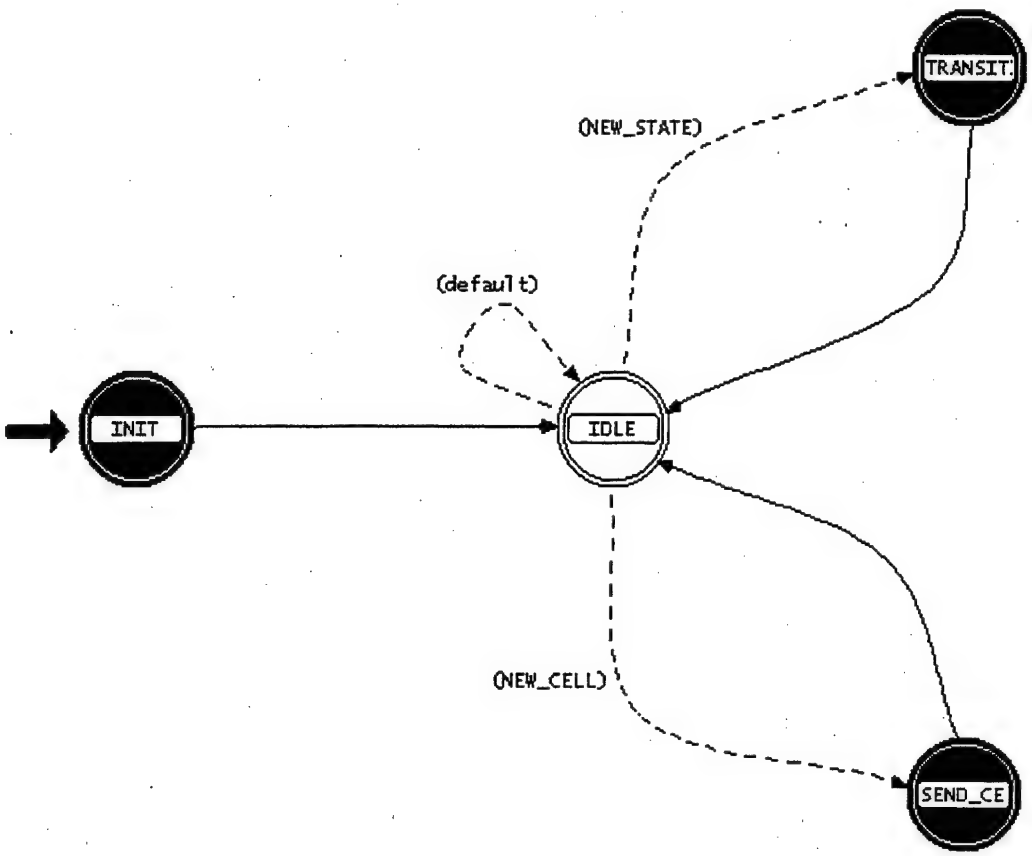

Figure A.2: Finite State Machine for a Layered Video Traffic Model.

1. Header Block

```
#include  "ams_pk_support.h"

#define SEND_CELL    0
#define CHANGE_STATE 100
#define MAX_SOURCE 7
#define MAX_LAYER  3

#define NEW_STATE  ((op_intrpt_type() == OPC_INTRPT_SELF) &&\
                    (op_intrpt_code() >= CHANGE_STATE))

#define NEW_CELL   ((op_intrpt_type() == OPC_INTRPT_SELF) &&\
                    ((op_intrpt_code() >= SEND_CELL) &&\
                     (op_intrpt_code() <= (SEND_CELL +
(MAX_SOURCE+1)*10))))

#define INF        9999999999
#define VCI_BASE   100
```

```
/* Event code = {Event type}{Source ID}{Layer ID} for 3 decimal digits.
*/

/* Cells are tagged by VCI = VCI + Source_ID
*/
/* Originating layer is indicating by the SDU and CLP bits.
*/

AtmT_Cell_Header_Fields* set_header(int,int);
```

2. State Variable Block

```
Objid       \self_id;
int         \curr_state[MAX_SOURCE];
int         \next_state[MAX_SOURCE];
Distribution**
            \state_dist;
double      \transit_time;
double      \interval;
Stathandle  \state0_shandle;
Stathandle  \state1_shandle;
Stathandle  \rate_shandle;
int         \sources;
Evhandle    \cell_intrpt[MAX_SOURCE];
int         \layer_state[MAX_SOURCE];
```

3. Temporary Variable Block

```
double M[6][6] = {{0.000,1.807,0.636,0.153,0.025,0.000},\
                  {1.240,0.000,0.288,0.399,0.044,0.022},\
                  {5.667,0.833,0.000,0.167,0.000,0.000},\
                  {2.800,3.920,0.280,0.000,0.000,0.000},\
                  {7.000,0.000,0.000,0.000,0.000,0.000},\
                  {0.000,7.000,0.000,0.000,0.000,0.000}};
double lambda[6] = {132.82,232.85,332.87,432.90,532.92,632.95};

Packet* cell_ptr;
int ix;
int jx;

int source_id;
int session_id;
int layer_id;

AtmT_Cell_Header_Fields* atm_hdr_ptr;
```

4. Function Block

set_header() creates an ATM cell header structure with the appropriate SDU- and CLP-bit tags for the layer.

```
AtmT_Cell_Header_Fields* set_header(int source_id,int layer_id)
{
   AtmT_Cell_Header_Fields* atm_hdr_ptr;

   //  Allocate memory for header fields.
   atm_hdr_ptr =
(AtmT_Cell_Header_Fields*)op_prg_mem_alloc(sizeof(AtmT_Cell_Header_Fiel
ds));

   // Load the VCI.
   atm_hdr_ptr->VCI = VCI_BASE + source_id;

   // Identify the layer.
   switch(layer_id){
      case 0:
         atm_hdr_ptr->PT = 0;
         atm_hdr_ptr->CLP = 0;
         break;
      case 1:
         atm_hdr_ptr->PT = 1;
         atm_hdr_ptr->CLP = 0;
         break;
      case 2:
         atm_hdr_ptr->PT = 0;
         atm_hdr_ptr->CLP = 1;
         break;
   }

   return atm_hdr_ptr;
}
```

5. Init State

The Init State creates an array of exponential distributions to represents transitions between states in the MMRP model. Each source is started arbitrarily in state 0.

```
/* get source module's own object id */
self_id = op_id_self();

/* get the requested number of multiplexed video sources */
op_ima_obj_attr_get (self_id, "Number_of_Sources", &sources);

/* allocate space and load distributions */
state_dist =
(Distribution**)(op_prg_mem_alloc(sizeof(Distribution*)*36));

for (ix=0;ix<6;ix++){
   for (jx=0;jx<6;jx++){
      if (M[ix][jx]>0.0){
```

```
            state_dist[ix*6+jx] =
op_dist_load("exponential",1.0/M[ix][jx],0);
         }
         else{
            state_dist[ix*6+jx] = op_dist_load("exponential",INF,0);
         }
      }
}

/* generate an initial interupt for each source, arbitrarily */
/* choosing the 0th state.                                    */
for (ix = 0;ix < sources;ix++){
   next_state[ix] = 0;
   op_intrpt_schedule_self(op_sim_time(),CHANGE_STATE + 10*ix);
}
```

6. Transition State

The Transition State reflects that a source is transitioning between states in the MMRP model. The time until the next transition is determined. The arrival rate for that source is updated to reflect the current state.

```
/* One of the sources is changing state; get the source's id.  */
session_id = op_intrpt_code() - CHANGE_STATE;
source_id = session_id/10;

/* Cancel the pending cell transmission self-interupt for this source.
*/
if (op_ev_valid(cell_intrpt[source_id])){
   op_ev_cancel(cell_intrpt[source_id]);
}

/* Assign the new current state.  */
curr_state[source_id] = next_state[source_id];

/* Find next state and transition time */
next_state[source_id] = 0;
transit_time = op_dist_outcome(state_dist[curr_state[source_id]*6]);

/* Search for the shortest time, this is the next state. */
for (ix = 1;ix < 6;ix++){

   interval = op_dist_outcome(state_dist[curr_state[source_id]*6 +
ix]);

   if (interval < transit_time){
      transit_time = interval;
      next_state[source_id] = ix;
   }
}
```

```
/* Reset the layer state counter. */
layer_state[source_id] = 0;
layer_id = layer_state[source_id];

/* Create a new formatted ATM cell. */
cell_ptr = op_pk_create_fmt("ams_atm_cell");

/* Allocate memory for the header and assign fields.  */
atm_hdr_ptr = set_header(source_id,layer_id);

/* ID the first cell of a GOB */
atm_hdr_ptr->GFC = 1;

/* Load the ATM header and transmit the cell. */
op_pk_nfd_set(cell_ptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
              op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));
op_pk_send(cell_ptr,0);
cell_intrpt[source_id] = op_intrpt_schedule_self(op_sim_time() +
1.0/lambda[curr_state[source_id]],\
                                    SEND_CELL + 10*source_id);

/* Schedule state transition */
op_intrpt_schedule_self(op_sim_time() + transit_time,CHANGE_STATE +
10*source_id);
```

7. Send_cell State

The Send_cell State transmits a new cell and schedules the next departure using the current arrival rate. In addition, the state determines the identity of the layer sending the cell. Bit allocation among layers, each layer's GOB length, and the manner of interleaving are all set here.

```
/* One of the sources is changing state; get the source's id.  */
session_id = op_intrpt_code() - SEND_CELL;
source_id = session_id/10;

/* Determine the layer id. */
layer_state[source_id] = (layer_state[source_id]++);
if (layer_state[source_id] > 7){
   layer_state[source_id] = 0;
}
switch(layer_state[source_id]){
   case 0: case 1: case 2: case 3:
      layer_id = 0;
      break;
   case 4: case 5:
      layer_id = 1;
```

```
      break;
   case 6: case 7:
      layer_id = 2;
      break;
}

/* Create and send an unformatted cell.  */
cell_ptr = op_pk_create_fmt("ams_atm_cell");

/* Allocate memory for the header and assign fields.  */
atm_hdr_ptr = set_header(source_id,layer_id);

/* ID the first cell of a GOB */
if (layer_state[source_id] == 0){
   atm_hdr_ptr->GFC = 1;
}
else{
   atm_hdr_ptr->GFC = 0;
}

/* Load the ATM header and transmit the cell. */
op_pk_nfd_set(cell_ptr,"header
fields",atm_hdr_ptr,op_prg_mem_copy_create,\
              op_prg_mem_free,sizeof(AtmT_Cell_Header_Fields));
op_pk_send(cell_ptr,0);

/* Schedule next cell departure. */
cell_intrpt[source_id] = op_intrpt_schedule_self(op_sim_time() +
1.0/lambda[curr_state[source_id]],\
                              SEND_CELL + 10*source_id);
```

APPENDIX B. MMRP MODEL PARAMETERS

The MMRP model parameters used in the OPNET simulations were developed using the procedure outlined in Section V.B. In accordance with the discussion presented in [90], the rate-controlled video trace shown in Figure B.1 was quantized to six levels. Table B.1 gives the state distribution vector calculated for the case of six states and the associated state arrival rates. Table B.2 gives the associated infinitesimal generating function for a frame rate of 10 fps.

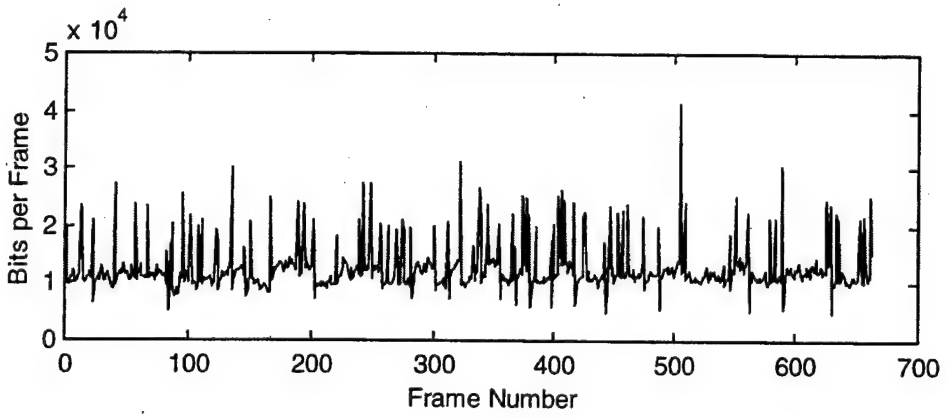

Figure B.1: Rate-controlled VBR Video Sequence.

State i	1	2	3	4	5	6
π_i	0.4136	0.4806	0.0618	0.0379	0.0045	0.0015
λ_i (cps)	132.82	232.85	332.87	432.90	532.92	632.95

Table B.1: State Probabilities and Arrival Rates for Quantized Video Source.

$$M = \begin{bmatrix} -2.6218 & 1.8073 & 0.6364 & 0.1527 & 0.0255 & 0 \\ 1.2405 & -1.9937 & 0.2880 & 0.3987 & 0.0443 & 0.0222 \\ 5.6667 & 0.8333 & -6.6667 & 0.1667 & 0 & 0 \\ 2.8000 & 3.9200 & 0.2800 & -7.0000 & 0 & 0 \\ 7.0000 & 0 & 0 & 0 & -7.0000 & 0 \\ 0 & 7.0000 & 0 & 0 & 0 & -7.0000 \end{bmatrix}$$

Table B.2: Infinitesimal Generating Function for Quantized Video Source.

For this sequence, six states give excellent results. Figure B.2 demonstrates how closely the MMRP captures the histogram of the original source. Mean bit rate is overpredicted but within 1% of the actual mean bit rate. Figure B.3 displays the autocorrelation function of both the model and the sequence, illustrating a close match over a period of 30 seconds. Figure B.3 also includes the model autocorrelation function when seven states are used. The closeness in tracking the autocorrelation function depends on how accurately the model predicts the mean bitrate. For this sequence, using 7 states gives a worse overprediction of the mean bitrate, thereby leading to the bias displayed in tracking the autocorrelation function. Increasing the number of states did not guarantee better results until a prohibitively large number of states were employed.

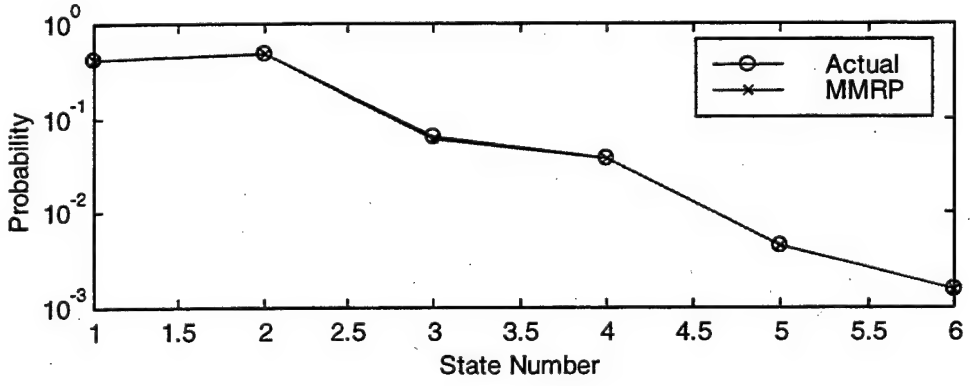

Figure B.2: Predicted and Actual Histograms.

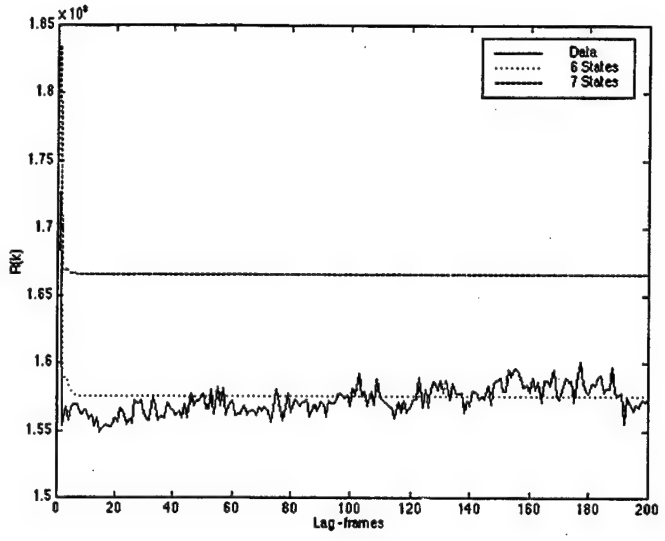

Figure B.3: Actual and Predicted Autocorrelation Functions.

LIST OF REFERENCES

[1] F. Kuo, W. Effelsberg, and J. Garcia-Luna-Aceves, *Multimedia Communications: Protocols and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1998.

[2] ADM A. Clemins, "IT-21: Information Technology for the 21st Century," presented at AFCEA West '98, San Diego, CA, Jan. 14-16, 1998.

[3] R. Schaphorst, *Videoconferencing and Videotelephony: Technology and Standards*, Norwood, MA: Aretech House, Inc., 1996.

[4] Chief of Naval Operations, N60 - Fleet and Allied C4 Requirements Division IT-21 Homepage, http://copernicus.hq.navy.mil/divisions/n6/n60/it21/.

[5] J. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh and R. Baker, *Digital Compression for Multimedia: Principles and Standards*, San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1998.

[6] K. Rao, J. Hwang, *Techniques & Standards for Image, Video & Audio Coding*, Upper Saddle River, NJ: Prentice-Hall, 1996.

[7] M. Riley and I. Richardson, *Digital Video Communications*, Norword, MA: Artech House, Inc., 1997.

[8] M. H. Willebeek-LeMair, Z. Shae, and Y. Chang, "Robust H.263 Video Coding for Transmission Over the Internet," IBM Res. Rep. RC20532, Aug. 1996.

[9] M. Willebeek-LeMair and Z. Shae, "Videoconferencing over Packet-Based Networks," *IEEE J. of Select. Areas in Comm.*, vol. 15, no. 6, pp. 1101-1114, 1997.

[10] RFC 1889, *RTP: A Transport Protocol for Real-time Applications*, Jan. 1996.

[11] D. Clark and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," *Proceedings, SIGCOMM '90, Computer Communication Review*, Sep. 1990.

[12] S. McCanne, M. Vetterli, and V. Jacobson, "Receiver-driven layered multicast," *Proc. SIGCOMM '96*, ACM, Stanford, CA, Aug. 1996.

[13] V. Rhee and J. D. Gibson, "Rate-Constrained Two-Layer Coding of H.261 Video," *28th Asilomar Conference on Signals, Systems, & Computers*, Pacific Grove, CA, 31 Oct - 2 Nov, 1994.

[14] P. Skelly, M. Schwartz, and M. Dixit, "A Histogram-Based Model for Video Traffic Behavior in an ATM Multiplexer," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 446-459, 8/1993.

[15] N. B. Shroff, "Traffic Modeling and Analysis in High Speed ATM Networks," Ph.D. Dissertation, Columbia University, New York, NY, 1995.

[16] W. Luo and M. Zarki, "Adaptive data partitioning for MPEG-2 video transmission over ATM networks," *Proc. IEEE Int'l Conf. Image Processing*, vol. 1, 1995.

[17] W. Stallings, *Data and Computer Communications*, Upper Saddle River, NJ: Prentice-Hall, 1997.

[18] W. Stallings, *High-Speed Networks: TCP/IP and ATM Design Principles*, Upper Saddle River, NJ: Prentice-Hall, 1998.

[19] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*, San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1996.

[20] RFC 1112, *Host Extensions for IP Multicasting*, Mar. 1992.

[21] M. Macedonia and D. Brutzman, "MBone provides audio and video across the Internet," *IEEE Computer*, pp. 30-36, Apr. 1994.

[22] RFC 1890, *RTP Profile for Audio and Video Conferences with Minimal Control*, Jan. 1996.

[23] The ATM Forum, *User-Network Interface Specification Ver. 3.1*, Upper Saddle River, NJ: Prentice-Hall, 1994.

[24] The ATM Forum Technical Committee, *622.08 Mbps Physical Layer Specification*, af-phy-0046.000, Jan. 1996.

[25] The ATM Forum Technical Committee, *Physical Interface Specification for 25.6Mb/s over Twisted Pair Cable*, af-phy-0040.000, Nov. 1995.

[26] The ATM Forum Technical Committee, *155.52 Mb/s Physical Layer Specification for Category-3 Unshielded Twisted Pair*, af-phy-0047.000, Nov. 1995.

[27] M. Schwartz, *Broadband Integrated Networks*, Upper Saddle River, NJ: Prentice-Hall, 1996.

[28] The ATM Forum Technical Committee, *Traffic Management Specification Version 4.0*, af-tm-0056.000, Apr. 1996.

[29] Recommendation I.121 - Broadband Aspects of ISDN, ITU-T, Geneva, April 1991.

[30] Recommendation I.362 - Adaptation Layer (AAL) Functional Description, ITU-T, Geneva 1992.

[31] Recommendation I.363 - B-ISDN ATM adaptation layer (AAL) specification, ITU-T, Geneva, Mar. 1993.

[32] B. Kercheval, *TCP/IP Over ATM: A No-Nonsense Internetworking Guide*, Upper Saddle River, NJ: Prentice-Hall, 1998.

[33] Recommendation I.363.2 - B-ISDN ATM Adaptation layer specification: Type 2 AAL, ITU-T, Geneva, Sep. 1997.

[34] J. Tsai, H. Fang, and C. Lee, "A Multicasting Solution for ATM Video Applications," *IEEE Trans. on Cricuits and Systems for Video Technology*, Vol. 7, No. 4, pp. 675-686, Aug. 1997.

[35] E. Gauthier, J. Le Boudec, and P. Oechslin, "SMART: A Many-to-Many Multicast Protocol for ATM," *IEEE Journal for Selected Areas in Comms.*, Vol. 15, No. 3, pp. 458-472, Apr. 1997.

[36] M. Kusayanagi et al., "ATM-based access network with multicast function for multimedia services," *Proc. SPIE*, Amsterdam, pp. 45-56, 1995.

[37] G. Armitage, "Multicast and multiprotocol support for ATM based internets," *ACM Sigcomm Comp. Commun. Rev.*, Vol. 25, No. 2, pp. 34-46, Apr. 95.

[38] R. Weekly, "Guaranteeing Fair Access for Multimedia Traffic Using the SMART Algorithm for Multicast ATM Connections," Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1999.

[39] A. Uziel, "Channel Allocation In Wireless Integrated Services Networks For Low-Bit-Rate Applications," Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, June 1998.

[40] S. Lapic and D. F. Gingras, "A priority-based random access spread spectrum protocol for integrated voice/data networks," *Proc. of the IEEE International Symposium on Circuits and Systems*, Monterey, CA, June 1-3, 1998.

[41] I. F. Akyildiz et al., "Medium Access Control Protocols for Multimedia Traffic in Wireless Networks," *IEEE Network Magazine*, pp. 39-47, July/August, 1999.

[42] S. Okubo et al. "ITU-T Standardization of Audiovisual Communication Systems in ATM and LAN Environments," *IEEE J. of Select. Areas in Comm.*, vol. 15, no. 6, pp. 965-982, 1997.

[43] Recommendation H.321 – Adaptation of H.320 visual telephone terminals to B-ISDN environments, ITU-T, Geneva, Feb. 1998.

[44] Recommendation H.310 – Broadband audiovisual communications systems and terminals, ITU-T, Geneva, Sep. 1998.

[45] S. McCanne, M. Vetterli , and V. Jacobson, "Low-Complexity Video Coding for Receiver-Driven Layered Multicast," *IEEE J. of Select. Areas in Comm.*, vol. 15, no. 6, pp. 983-1001, 1997.

[46] M. Krunz, "Bandwidth Allocation Strategies for Transporting Variable-Bit-Rate Video Traffic," *IEEE Comms. Magazine*, Vol. 37, No. 1, pp. 40-46, January 1999.

[47] The ATM Forum Technical Committee, *UNI Signaling 4.0*, af-sig-0061.000, Jul. 1996.

[48] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Upper Saddle River, NJ: Prentice-Hall, 1995.

[49] C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Upper Saddle River, NJ: Prentice-Hall, 1992.

[50] Z. Xiong, K. Ramchandran, M. T. Orchard, and Y. Zhang, "A comparative study of DCT and wavelet based coding," *Proc. of the IEEE International Symposium on Circuits and Systems*, Monterey, CA, June 1-3, 1998.

[51] T. Eude et al., "On the distribution of DCT coefficients," *ICASSP '94*, vol. 5, pp. 365-368, Adelaide, Australia, Apr. 1994.

[52] R. Reininger and J. Gibson, "Distribution of two dimensional DCT coefficents for images," *IEEE Trans. Commun.*, Vol. COM-31, pp. 835-839, June 1983.

[53] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, Vol. 15, No. 6, pp. 74-90, Nov. 1998.

[54] R. Frederick, "Experiences with real-time software video compression," *Proc. 6th Int. Workshop Packet Video*, Portland, OR, Sept. 1994.

[55] Telenor Research, *Video Codec Test Model, Tmn5*, Jan. 1995.

[56] ITU-T Recommendation H.263, *Video Coding for Low Bitrate Communication*, 1996.

[57] N. Tanabe and N. Farvardin, "Subband image coding using entropy-coded quantization over noisy channels," *IEEE Journal on Sel. Areas in Commun.*, Vol. 10, No. 5, pp. 926-942, Jun 1992.

[58] T. Senoo and B. Girod, "Vector quantization for entropy coding of image subbands," *IEEE Trans. on Image Proc.*, Vol. 1, No. 4, pp. 526-532, Oct. 1992.

[59] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Sig. Proc., Special Issue on Wavelets and Signal Processing*, Vol. 41, No. 12, pp. 3445-3462, Dec. 1993.

[60] G. Kaiser, "The Fast Haar Transform," *IEEE Potentials*, vol. 17, no. 2, pp. 34-36, 1998.

[61] J. Bolot and T. Turletti., "A Rate Control Mechanism for Packet Video in the Internet," *IEEE INFOCOM'94*.

[62] T. Sakatani, "Congestion avoidance for video over IP network," *Int. COST 237 Workshop Proc.*

[63] H. Kanakia et al., "An adaptive congestion control scheme for real-time packet video transport," *IEEE/ACM Trans. Networking*, vol. 3, no. 12, pp. 671-682, Dec. 1995.

[64] S. Chakrabarti and R. Wang, "Adaptive control for packet video," *IEEE Multimedia '94*, pp. 56-62.

[65] Y. Chung, J. Kim, and C.-C. Kuo, "Real-time Streaming Video with Adaptive Bandwidth Control and DCT-based Error Concealment," *Trans. ISCAS'98*, Monterey, CA, 1998.

[66] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.

[67] ITU-T Recommendation H.261, *Video Coding for Low Bitrate Communication*, 1996.

[68] V. Rhee and J. D. Gibson, "Block-Level Refinement of Motion Description in Layered H.261 Video," *Proc. 29th Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 29 Oct - 1 Nov 1995.

[69] Burt, P. J. and Adelson, E. H., "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532-540, 1983.

[70] Brown, T. B., Cantrell, P. E., and Gibson, J. D., "Multicast Layered Video Teleconferencing: Overcoming Bandwidth Heterogeneity," *1996 Int. Conf. on Image Processing*, Lausanne, Switzerland.

[71] H. Gharavi and M. H. Partovi, "Video coding and distribution over ATM for multipoint teleconferencing," *Proc. GLOBECOM '93*, Houston, TX, Dec. 1993.

[72] P. Bahl and W. Hsu, "Adaptive Region-Based Multi-Scaled Motion-Compensated Video Coding for Error Prone, Bandwidth-Limited Communication Channels," *Proc. Conf. Broadband Networking Technologies, SPIE's Int'l Symp. Voice, Video and Data Communications*, Dallas, TX, Nov. 1997.

[73] C. Diab, R. Prost, and R. Goutte, "Block-Adaptive Subband Coding of Images," *ICASSP*, 1990, pp. 2093-2096.

[74] S. Skretkowicz, "Implementing of a Low-Complexity, Adaptive, Layered Video Coder for Video Teleconferencing," Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1999.

[75] ITU-T Recommendation T.81, *Information Technology - Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines*, September 1992.

[76] J. Choi and D. Park, "A stable feedback control of the buffer state using the controlled lagrangian multiplier method," *IEEE Trans. on Image Proc.*, vol. 3, no. 5, pp. 546-558, Sept. 1994.

[77] J. Ribas-Corbera and S. Lei, "Rate control for low-delay video communications," ITU-T SG16/Q15 document Q15-A-10, June 1997.

[78] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[79] J. Ribas-Corbera and S. Lei, "Rate control in DCT video coding for low-delay video communications," *IEEE Trans. On Circuits and Systems for Video Tech.*, Nov. 1998.

[80] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech and Signal Proc.*, Vol. 36, No. 9, pp. 1445-1453, Sept. 1988.

[81] I-Ming Pao and Ming-Ting Sun, "A Rate-Control Scheme for Streaming Video Encoding," *Proc. of 32nd Asilomar Conference on Signals, Systems, & Computers*, Pacific Grove, CA, 31 Oct - 2 Nov, 1998.

[82] R. O. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues*, Artech House Inc., Norwood, MA, 1994.

[83] J. S. Turner, "New Directions in Communications (or Which Way in the Information Age?)," *IEEE Commun. Mag.*, Vol. 24, No. 10, pp. 8-15, Oct. 1986.

[84] E. P. Rathgeb, "Modeling and Performance Comparison of Policing Mechanisms for ATM Networks," *IEEE JSAC*, Vol. 9, No. 3, pp. 325-334, April 1991.

[85] M. W. Garret and W. Willinger, "Analysis, Modeling, and Generation of Self-Similar VBR Video Traffic," *Proc. SIGCOMM '94*, London, pp. 269-280, Aug. 1994.

[86] B. Malglaris, D. Anastassiou, P. Sen, and G. Karlsson, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Comm.*, Vol. 36, No. 7, pp. 834-843, 1988.

[87] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM networks," *IEEE Trans. Ckts. And Systems*, Vol. 2, No. 1, pp. 49-59, 1992.

[88] P. Sen, N. Rikli, and D. Anastassiou, "Models for packet switching of variable bit-rate video sources," *IEEE J. on Select. Areas in Comm.*, Vol. 7, No. 5, pp. 865-869, June 1989.

[89] S. S. Dixit and P. Skelly, "Video Traffic Smoothing and ATM Multiplexer Performance," *Proc. IEEE Globecom '91*, Phoenix, AZ, Dec. 1991.

[90] R. Parker and M. Tummala, "Modeling of H.263 Encoded Low Bitrate Video Traffic for Tactical Video Conferencing Applications," *Proc. of 32nd Asilomar Conference on Signals, Systems, & Computers*, Pacific Grove, CA, 31 Oct - 2 Nov, 1998.

[91] A. I. Elwalid and D. Mitra, "Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 329-343, June 1993.

[92] N. B. Shroff and M. Schwartz, "Improved Loss Calculations at an ATM Multiplexer," *IEEE/ACM Trans. On Networking*, Vol. 6, No.4, pp. 411-421, Aug. 1998.

[93] R. Parker and M. Tummala, "A Scheduling Algorithm for Layered Video Traffic with QoS Guarantees," *OPNETWORLD '99*, Washington, DC, 30 August – 03 September, 1999.

[94] R. Parker and M. Tummala, "Modeling of Layered Video for Low-Bit-Rate Tactical Video Conferencing Applications," *ISCAS '99*, Orlando, FL, 30 May – 02 June, 1999.

[95] T. S. Randhawa and R. H. S. Hardy, "Estimation and Prediction of VBR Traffic in High-Speed Networks using LMS Filters," *Trans. IEEE International Conference on Communications*, Atlanta, GA, June 1998.

[96] S. Haykin, *Adaptive Filter Theory*, 3rd Ed., Prentice Hall, Upper Saddle River, NJ, 1996.

[97] S. C. Liew and H. H. Chan, "Lossless Aggregation: A Scheme for Transmitting Multiple Stored VBR Video Streams over a Shared Communications Channel Without Loss of Image Quality," *IEEE J. of Select. Areas in Comm.*, Vol. 15, No. 6, pp. 1181-1189, 1997.

[98] K. Sriram, "Dynamic Bandwidth Allocation and Congestion Control Schemes for Voice and Data Multiplexing in Wideband Packet Technology," *Trans. IEEE International Conference on Communications*, 1990, pp. 1003-1009.

[99] S. S. Panwar, D. Towsley, and J. K. Wolf, "Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Service," *J. of the ACM*, vol. 35, no. 4, pp. 832-844, Oct. 1988.

[100] G. Kuo and P. Ko, "Achieveing Minimum Slice Loss for Real-Time MPEG-2-Based Video Networking in a Flow-Oriented Input-Queued ATM Switching

Router System," *IEEE Comms. Magazine*, Vol. 37, No. 1, pp. 58-61, January 1999.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Prof. Murali Tummala, Code EC/Tu 4
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Prof. Roberto Cristi, Code EC/Cx 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Prof. David C. Jenn, Code EC/Jn 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Prof. John E. McEachen, Code EC/Mj 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

8. Prof. C. Thomas Wu, Code CS/Wu 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5118

9. Dr. Don Gingras 1
SPAWAR Systems Center San Diego, Code D8805
Communication and Information Systems Department
San Diego, CA 92152-5001

10. Dr. Richard C. North 1
SPAWAR Systems Center San Diego, Code D885
53560 Hull Street
San Diego, CA 92152-5001

11. Commandant (G-ADW) 1
US Coast Guard
Room 5514
2100 Second Street, SW
Washington, DC 20593-0001
Attn: LCDR John D. Wood

12. LT Howard Pace Jr. 1
Spawar Systems Center San Diego, Code D855
53560 Hull Street
San Diego, CA 92152-5001

13. LCDR Robert E. Parker, U.S. Navy 2
2452 Ponte Vedra Way
Chula Vista, CA 91915

14. LT Steven J. Skretkowicz, U.S. Navy 1
363A Shark Blvd.
Groton, CT 06340

15. LCDR Randolph R. Weekly, U.S. Navy 1
315 Arloncourt Rd
Seaside, CA 93955